VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

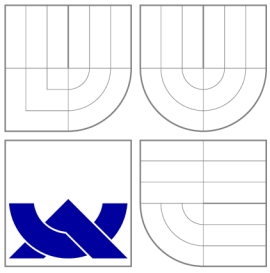
INTELIGENTNÍ KLIENT PRO HUDEBNÍ PŘEHRÁVACÍ SERVER MPD

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

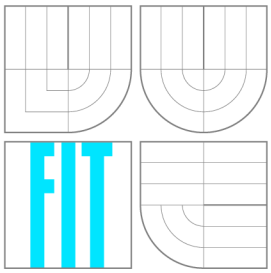
AUTOR PRÁCE
AUTHOR

Bc. TOMÁŠ WAGNER

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

INTELIGENTNÍ KLIENT PRO HUDEBNÍ PŘEHRÁVACÍ SERVER MPD

INTELLIGENT CLIENT FOR MUSIC PLAYER DAEMON

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. TOMÁŠ WAGNER
AUTHOR

VEDOUCÍ PRÁCE — Doc. Ing. VLADIMÍR JANOUŠEK, Ph.D.
SUPERVISOR

BRNO 2012

Abstrakt

Cílem diplomové práce je návrh a implementace inteligentní klientské aplikace pro hudební přehrávací server Music Player Daemon (MPD), která vyhledává a prezentuje související metadata k přehrávanému obsahu. Samotnému návrhu předchází teoretický rozbor, který zahrnuje analýzu agentních systémů, metod klasifikace dat, webových komunikačních protokolů a jazyků pro popis dokumentu. Současně je proveden rozbor MPD serveru a komunikačního protokolu využívaného klientskou aplikací. Dále jsou v práci popsány současné klientské aplikace, které prezentují uživateli metadata. V posledních kapitolách práce je popsán návrh a implementace klientské a serverové aplikace. Je rozebrán způsob řešení implementace a řešení vzniklých problémů. V práci jsou také popsány výsledky testování.

Abstract

The content of this master thesis project is about design and implementation of intelligent client application for Music Player Daemon (MPD), which searches and presents the metadata related to played content. The actual design precedes the theoretical analysis, which includes analysis of agent systems, methods of data classification, web communication protocols and languages for describing HTML document. At the same time is analyzed the MPD server and communication protocol used by clients application. Furthermore, this work describes the current client applications that presents metadata. In the last chapters of the thesis describes the design and implementation of intelligent client. It describes the methods of solution the implementation and solution of problems. Lastest chapters describes the testing result.

Klíčová slova

MPD klient, multiagentní systém, klasifikace dat, autonomní systém, inteligentní klient, metadata

Keywords

MPD client, multiagent system, data classification, autonomy system, intelligent client, metadata

Citace

Tomáš Wagner: Inteligentní klient pro hudební přehrávací server MPD, diplomová práce, Brno, FIT VUT v Brně, 2012

Inteligentní klient pro hudební přehrávací server MPD

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Doc. Ing. Vladimíra Janouška, Ph.D.

.....
Tomáš Wagner
22. května 2012

Poděkování

Rád bych touto cestou poděkoval panu docentu Vladimíru Janouškovi za pomocné korekce, rady a nápady při řešení diplomové práce. Dále bych rád poděkoval rodině a přítelkyni Dorotě za jejich morální podporu.

Obsah

Kapitola 1

Úvod

S rozvojem internetu, webových služeb a informačních zdrojů, které nabízí se v dnešní době objevují i možnosti automatizovaného využití k získávání souvisejících informací. Informací, které jsou aktuální, zajímavé a souvisí s konkrétními požadavky uživatele. Jde o informace často informačně hodnotné, jelikož jsou aktualizované v čase, rozvíjejí se, vytváří se a nestárnou.

Téma této práce se zaobírá teoretickým rozborem, návrhem a implementací inteligentního klienta pro Music Player Daemon (MPD). Inteligentní klient pracuje v prostředí internetu a webových služeb, které internet poskytuje. Z tohoto prostředí pak získává informace, tzv. metadata, která prezentuje uživateli. Jelikož se jedná o klientskou aplikaci pro hudební přehrávací server, prezentované informace souvisí s aktuálně přehrávanou skladbou. Klientská aplikace dále zahrnuje inteligenci, kdy využívá informací od uživatele z hodnocení prezentovaného obsahu. Z uživatelského hodnocení informací se postupně učí a rozvíjí tak své schopnosti prezentovat relevantní informace.

V kapitole č. 2 je proveden teoretický rozbor, kde popíši agentní systémy. Zde uvedu popis reaktivního agenta, cílem řízeného agenta až po učícího se agenta. Učící se agent je pro tuto práci stěžejní a jeho činnost je rozebrána detailněji. Důležitou částí pro schopnost učení a inteligence jsou také metody klasifikace. V této podkapitole uvedu klasifikační algoritmy pro roztřídění textových dat do tříd se stejnými vlastnostmi. Provedu rozbor rozhodovacích stromů, klasifikace na základě pravidel, Bayesovské klasifikace, metod založených na vzdálenosti a neuronových sítí. V dalších dvou podkapitolách uvedu rozbor komunikačních protokolů a jazyků pro popis dokumentů, využívaných v současném webovém prostředí. Zde uvedu např. jazyk HTML, XML, pro něž existuje DOM model, což je rozhraní, prostřednictvím kterého lze přistupovat k jednotlivým prvkům dokumentu. Posledním jazykem pro popis dokumentu je jazyk JSON, což je minimalistický textový formát používaný především pro svou jednoduchost a rychlost interpretace. Poslední kapitolou teoretického rozboru je kapitola popisující serverovou část aplikace Music Player Daemon. V této kapitole se zaměřím především na princip komunikace se serverem. Na příkazy, které server akceptuje a autentizační možnosti.

V kapitole č. 3 představím několik současných klientských aplikací. Jedná se zejména o aplikace, které určitým způsobem prezentují metadata uživateli. První z představovaných aplikací bude konzolová aplikace NCMPCPP. Ta využívá pouze konzolového přístupu jak pro ovládání, tak pro prezentování metadat. Dalšími aplikacemi jsou již aplikace s grafickým uživatelským rozhraním, jako je např. Sonata, Gnome Music Player Client a další. V těchto aplikacích se již objevují rozsáhlejší prezentace metadat. Obsahující např. informace o interpretovi, obrázky interpretů, obaly alb, texty skladeb atp.

V kapitole č. 4 uvedu návrh implementace pro aplikaci inteligentního klienta. Ukáži schéma modulů, ze kterých se inteligentní klient skládá a jak by jednotlivé moduly měly fungovat. Dále definuji zdroje informací, které rozdělím na primární a sekundární dle ověřených a neověřených zdrojů. Uvedu také možnosti, jak zjišťovat informace z obsahu webového prostředí, a představu, jak urychlit prohledávání na daném webu. Součástí návrhu je také návrh grafického uživatelského rozhraní, které z velké části obsahuje prostor pro prezentaci získaných metadat a relevantních informací k přehrávané skladbě.

V kapitole č. 5 představím způsob řešení implementace. V úvodu kapitoly uvedu použité knihovny, které ulehčily implementaci. Uvedu důvody, které vedly k využití vybraných knihoven. Dále představím uživatelské rozhraní programu, jeho řešení pro maximální možné využití místa pro prezentaci a možnosti ovlivňování zobrazovaných dat. V závěru kapitoly představím implementovanou funkčnost ať už možnosti prezentace dat, hodnocení a klasifikace či možnosti sdílení. Nakonec je rozebráno zpracování dat. Jedná se o zpracování textových a obrazových informací, zpracování průchodů webovými stránkami a ukládání dat.

V kapitole č. 6 se zaměřím na implementaci serverové části, která je prostředkem pro možnosti sdílení dat mezi jednotlivými klientskými aplikacemi.

V kapitole č. 7 uvedu tři druhy testů. První test obsahuje měření úspěšnosti klasifikace ohodnocených záznamů. Druhý test zahrnuje testování učících schopností pro možnosti zdokonalování a poslední závěrečný test představí výkonnostní měření.

1.1 Značení v textu

V textu bude použito tučného zvýraznění písma u důležitých tvrzení. Příklady zdrojových kódů v textu budou napsány neproporcionálním písmem.

Kapitola 2

Teoretický rozbor

2.1 Agentní systémy

Představitelem agentního systému je inteligentní autonomní agent. Jedná se o samostatnou entitu, která je umístěna v konkrétním prostředí a vykonává danou činnost [14]. Autonomnost je charakteristická tím, že agent je schopen pracovat samostatně bez vnějších zásahů, řídit své úkony a kontrolovat vnitřní stav ve kterém se nachází. V této práci se zaměřím hlavně na prostředí webu. Prostředí webu je dynamické. Vyvíjí a mění se v čase, přičemž velikost prostředí je nemožné sledovat jedním agentem. Agent má tedy k dispozici k prohledávání pouze část prostředí. Každý agent se skládá ze senzorů (tzv. perceptorů) a akčních členů (tzv. aktuátorů) pro provádění diskrétních akcí do prostředí. Důležitou vlastností je také udržování báze znalostí, která agentovi pomáhá se rozhodovat při výběru akce.

Každý agent provádí činnost. Dokud je nespokojen s výsledkem, sleduje prostředí, aktualizuje bázi znalostí a dle znalostní báze vybírá a provádí akce. Kooperace mezi agenty je žádoucí. V takovém případě se jedná o multiagentní systém. Agenti si mezi sebou předávají informace, které pomáhají v rozhodování při výběru akce. Pro komunikaci může být zvolen v podstatě libovolný komunikační protokol. V případě realizace multiagentního systému v rámci jedné aplikace to může být např. zasílání zpráv mezi objekty. V jiných případech může být použit libovolný protokol (např. HTTP) nebo speciální protokol ACL (Agent Communication Language).

2.1.1 Reaktivní agent

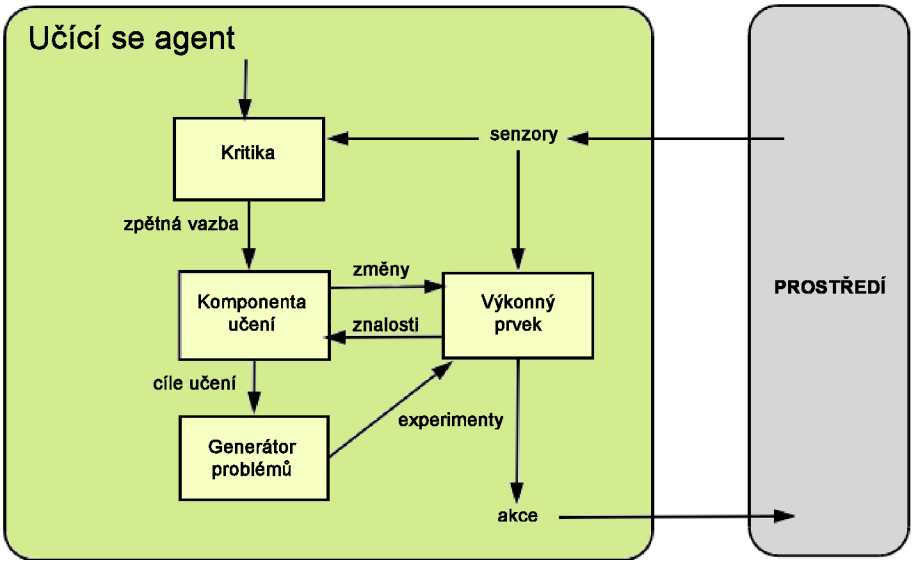
Nejjednodušší forma agenta, agent reaguje na základě aktuálního vjemu a přitom nesleduje žádné akce, které již provedl dříve. Celá funkce agenta je založena pouze na podmínkách: „Pokud tato podmínka platí, provedu následující akci“. Často se ale může dostat do nekonečných smyček, ve kterých se zacyklí. Inteligentnější verzí reaktivního agenta je totožný agent doplněný o model. Ve své rozhodovací logice má uložený model konkrétního prostředí. Současný stav je daný aktuálním stavem modelu a stavem z prostředí.

2.1.2 Cílem řízený agent

Cílem řízení agenti dále rozšiřují možnosti reaktivních agentů s modelem. Pomocí stanoveného cíle. Cílová informace popisuje situaci, která by měla nastat a je situací žádoucí. To umožňuje agentovi určit cestu, kterou se má dále ubírat při prohledávání stavového prostoru.

2.1.3 Učící se agent

Nejvíce inteligentních schopností ze všech modelů umožňuje dosáhnout model agenta s prvkem učení. Výhodou je možnost pracovat v neznámém prostředí, kdy se v tomto prostředí může stát daleko schopnější než jsou jeho počáteční znalosti. Na obr.2.1 je uvedena struktura učícího se agenta, která obsahuje čtyři komponenty. Komponenta učení provádí učení a vylepšení agentova rozhodování. Komponenta výkonného prvku je zodpovědná za výběr externí akce. Učení probíhá učením kritikou. Na základě informace z vnějšího prostředí, rozhoduje, jaké akce výkonný prvek provede aby se výsledek do budoucna vylepšoval. Poslední komponentou je generátor problémů. Tato komponenta je z hlediska učícího se agenta důležitá. Umožní agentovi vygenerovat jiný výsledek, který by ho mohl přivést k nové zkušenosti, ze které by se poučil.



Obrázek 2.1: Struktura učícího se agenta. Převzato z [14]

2.2 Metody klasifikace

Klasifikace umožňuje zařadit záznamy z množiny $Z = \{z_1, z_2, ..., z_n\}$ do tříd z množiny $C = \{C_1, C_2, ..., C_m\}$ a představuje tedy zobrazení $f : Z \to C$, kdy se předpokládá, že každý záznam z_i spadá právě do jedné z tříd. Dochází tedy k třídění dat založené na jejich informačním obsahu. Záznam je dán množinou atributů $A = \{A_1, A_2, ..., A_o\}$ kde hodnota atributu $A_i = \{a_{i1}, a_{i2}, ..., a_{ip}\}$. Pro třídění se využívá několik druhů algoritmů, které v dalších podkapitolách jednotlivě popíši.

Proces klasifikace se skládá ze dvou kroků. Prvním krokem je učení, kdy se vytváří klasifikační model a doplňují se trénovací záznamy. Druhým krokem je vlastní klasifikace, kdy se na základě trénovacích záznamů provádí klasifikace nových záznamů do tříd. Většina klasifikačních algoritmů je založena na teorii pravděpodobnosti a matematické statistice.

Hodnocení klasifikace z hlediska kvality je důležité pro ohodnocení každého z algoritmů. Pro hodnocení se zavádí míry jako je citlivost, specifičnost a přesnost. Označím-li A počet záznamů, které klasifikátor správně zařadil do třídy C. B jako počet záznamů, které byly do třídy C zařazeny nesprávně. C jako počet záznamů patřících do třídy C, které klasifikátor

do této třídy nezařadil a D jako počet záznamů nepatřících do třídy C, které klasifikátor nezařadil do třídy C. Podle zavedeného označení lze potom odvodit následující vzorce:

$$citlivost = \frac{A}{A+C} \tag{2.1}$$

$$specificnost = \frac{D}{B+D} \tag{2.2}$$

$$presnost = \frac{A}{A+B} \tag{2.3}$$

Mezi další kritéria hodnocení klasifikace může patřit:

- Rychlost - závisí na ceně výpočtů při použití vybraného klasifikačního algoritmu,
- Robustnost - je schopnost třídění klasifikačního algoritmu a tvorba správných předpovědí,
- Škálovatelnost - schopnost práce klasifikačního algoritmu při třídění vzhledem k velkému množství dat,
- Interpretovatelnost - především úroveň porozumění a pochopení výsledků vybraného klasifikátoru, je velmi subjektivní.

2.2.1 Rozhodovací stromy

Rozhodovací stromy jsou obecně n-ární stromy, kde uzly reprezentují hodnoty vybraného atributu a koncové uzly (listy) reprezentují třídy. Obecně jsou jednou z často využívaných klasifikačních technik. Mezi výhody patří snadná implementace, efektivní použitelnost. Umožňují generování pravidel, které lze snadno interpretovat a je tak patrný jejich význam. Spadají do kategorie algoritmů učení s učitelem. Používají se, je-li počet tříd, do kterých klasifikaci provádíme předem znám a není příliš velký.

Jedním z algoritmů využívající rozhodovací stromy je algoritmus ID3, jehož činnost rozepíši detailněji. Algoritmus ID3 provádí rozvětvování na základě výběru atributu s nejvyšším informačním ziskem. Ten je vypočitatelný pomocí pravděpodobnosti uvedeném vzorcem 2.4 [10]. K již definovaným množinám nyní zavedu množinu $S = \{S_1, S_2, ..., S_m\}$. Tato množina obsahuje množinu záznamů rozdělených dle příslušnosti k třídě. Prvky množiny S_i jsou tedy množiny záznamů rozdělené dle příslušnosti k jedné z tříd.

$$Gain(A_x) = Info(S) - Entrophy(A_x) \tag{2.4}$$

$$Info(S) = -\sum_{i=1}^{m} p_i log_2(p_i) = \sum_{i=1}^{m} p_i log_2(\frac{1}{p_i}) \tag{2.5}$$

kde p_i je pravděpodobnost, že množina záznamů S_i patří do třídy C_i. Logaritmus o základu 2 je použit kvůli tomu, že výsledek informace je zakódován v bitech. Výsledek pak udává množství informace potřebné k identifikaci třídy množiny S. Dále definuji entropii [10], která je založena na rozvětvení dle A_x. Udává tak míru informační hodnoty atributů.

$$Entrophy(A_x) = \sum_{j=1}^{v} \frac{|S_j|}{|S|} \cdot Info(S_j) \tag{2.6}$$

Rozšířením algoritmu ID3 je algoritmus C4.5, který vylepšuje práci s chybějícími daty. Kdy chybějící data jsou ignorována při výpočtu informačního zisku a při klasifikaci jsou predikována vzhledem k ostatním hodnotám stejného atributu. Dále pak algoritmus umožňuje ořezávání, dělení stromů a využití spojitých dat v klasifikaci, která jsou uvedena ve tvaru intervalů.

2.2.2 Bayesovská klasifikace

Statistická metoda učení, která je široce používána pro svojí jednoduchost, rychlost učení, použití a aktualizaci [5, s.148]. Využívá se např. jako filtr nevyžádaných emailových zpráv, který se dle uživatelových zásahů učí lépe rozpoznávat nevyžádané zprávy a označovat je. Je určována dle Bayesova teorému 2.7[10, s.311].

$$P(H|X) = \frac{P(X|H)\,P(H)}{P(X)} \tag{2.7}$$

kde X je záznam (obecně n-tice), jehož třídu zatím neznáme, H je hypotéza, že X patří do určité třídy C. $P(H|X)$ se nazývá posteriorní pravděpodobnost (posterior probability) platnosti H pro záznam X. $P(H)$ je priorní pravděpodobnost (prior probability) hypotézy H. $P(X|H)$ je posteriorní pravděpodobnost, že se jedná o X, platí-li H.

Naivní Bayesovská klasifikace vychází z předpokladu, že jednotlivé atributy záznamu jsou na sobě nezávislé. Nechť $Z_1, Z_2, ..., Z_n$ jsou záznamy, $C_1, C_2, ..., C_m$ třídy a máme záznam $X = (x_1, x_2, ..., x_n)$. X bude přiřazen třídě s maximální posteriorní pravděpodobností. Tedy $P(C_i|X) > P(C_j|X)$ pro $1 \leq j \leq m$ a zároveň $i \neq j$. $P(C_i|X)$ se vypočítá dle Bayesova teorému jako

$$P(C_i|X) = \frac{P(X|C_i)\,P(C_i)}{P(X)} \tag{2.8}$$

kde $P(C_i) = \frac{nc_i}{nc}$, kde $nc_i = |C_i|$ je počet trénovacích záznamů ze třídy C_i a $nc = |C|$ je celkový počet záznamů třídy C. $P(X)$ je konstanta, kterou ve výpočtu není nutné uvažovat. Za předpokladu nezávislosti položek záznamů pak platí 2.9.

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \tag{2.9}$$

2.2.3 Metody založené na vzdálenosti

Metody klasifikátorů vzdálenosti od nejbližších sousedů jsou založeny na porovnávání testovací množiny s trénovací množinou dat a jejich podobností. Množina trénovacích dat obsahuje n atributů. Každá množina pak reprezentuje bod v n dimenzionálním prostoru. Jedním z algoritmů je k-Nearest Neighbor klasifikátor, který zařadí zkoumaný záznam do třídy, která přísluší většině z K nejbližších záznamů trénovací množiny dat. Algoritmus je velmi citlivý na volbu K sousedů. Doporučená volba $K = \sqrt{pocet\_trenovacch\_dat}$. Pro výpočet se využívá Eukleidovská vzdálenost v n dimenzionálním prostoru mezi body X_1 a X_2 dle 2.10[10].

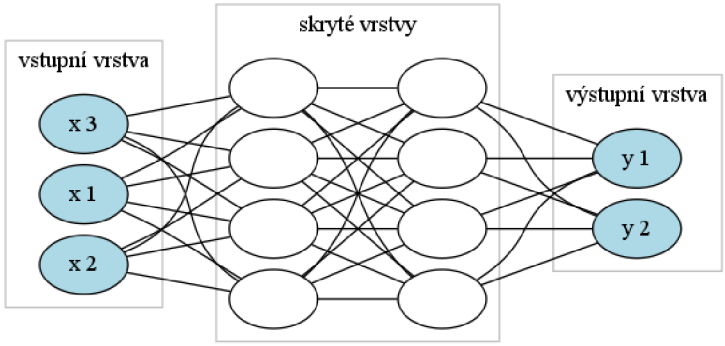
$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n}(x_{1i} - x_{2i})^2} \tag{2.10}$$

Největším problémem metod, využívajících vzdálenosti, je návrh a implementace vhodné metriky. Jelikož platí, že metriky jsou často závislé na druhu záznamů.

2.2.4 Neuronové sítě

Neuronové sítě vytváří přístup inspirovaný přírodou. Mají široké možnosti uplatnění. Zde se zaměřím pouze na uplatnění z hlediska klasifikace dat. Obecně neuronové sítě vykazují dlouhé časy učení a proto jsou využívány především v aplikacích, kde nám tento fakt vyhovuje. V souvislosti s touto prací jsou neuronové sítě nevyhovujícím modelem. Často jsou také kritizovány pro svoji neinterpretovatelnost pro člověka.

Neuronovou sítí se rozumí neorientovaný graf s množinou uzlů a hran, kdy uzly jsou členěny do vrstev. První vrstvou je vstupní vrstva a poslední výstupní vrstva. Mezi nimi se nachází skryté vrstvy. Neurony jsou spojené pouze dopředně a to směrem od vstupu k výstupu. Každá hrana mezi uzly má přiřazenou váhu. Ilustrační příklad neuronové sítě bez vah jsem uvedl na obr.2.2



Obrázek 2.2: Příklad neuronové sítě a jejích vrstev převzato z [8].

Klasifikace a učení neuronové sítě pak probíhá např. s využitím algoritmu zpětné propagace(Backpropagation). Ten iterativně zpracovává množinu trénovacích záznamů a pro každý záznam porovnává předpovědi sítě s cílovým záznamem. Pro každý trénovací záznam se pak modifikují váhy a proto dochází k minimalizaci chyby.

2.3 Webové komunikační protokoly

V současnosti je **HTTP (HyperText Transfer Protocol)** nejvyužívanějším protokolem na internetu. Jedná se o bezstavový protokol určený např. pro výměnu dokumentů ve formátu HTML. Standardní verze protokolu HTTP ve verzi 1.1 definovaná dle RFC2616[11] je protokolem, který nevyužívá šifrovaného spojení a přenášená data jsou tak čitelná pro případného útočníka, který komunikaci zachytí. Proto byl z důvodů bezpečnosti definován další protokol HTTPS (HyperText Transfer Protocol Secure) dle RFC2818[12]. Protokoly HTTP a HTTPS jsou využity serverem, poskytujícím dokumenty. Typickým příkladem může být server Apache. Protokol HTTP je používán nejčastěji na portu číslo 80, HTTPS protokol je používán nejčastěji na portu 443.

Pro návrh inteligentního klienta je důležitý fakt, že je protokol bezstavový. Tudíž server neudržuje kontext mezi jednotlivými požadavky klienta. Protokol je typu požadavek-odpověď, kdy klient zasílá požadavek na server, server jej zpracuje a vrací zpět klientovi odpověď. Požadavek je možné na server zaslat sedmi možnými metodami:

- metodou GET,

- metodou POST,
- metodou HEAD,
- metodou OPTIONS,
- metodou PUT,
- metodou DELETE,
- metodou TRACE.

Metoda GET společně s metodou POST patří mezi nejpoužívanější metody. Hlavním rozdílem metody GET od metody POST je délka požadavku zasílaného na server. U metody GET je serverem standardně nastaven limit 8kB na požadavek, u metody POST standardně 8MB. Limity mohou být dle implementace serveru rozdílné.

Využití metody GET je především k dotazování na konkrétní dokument. POST se využívá hlavně při odesílání objemnějších dat na server, jako jsou např. formuláře. Mezi další metody patří metoda HEAD, je podobná metodě GET, avšak odpověď ze serveru se skládá pouze z hlavičky. Tím lze ušetřit čas a šířku přenosového pásma při dotazování na server, zda se např. daný dokument nemodifikoval. Metoda OPTIONS se používá na dotaz o možnostech serveru. Metoda PUT funguje podobně jako metoda GET, ale uchovává tělo požadavku na místě daném dle požadované URL. Poslední metody se již v běžných implementacích aplikací nevyužívají. Jedná se o metody DELETE a TRACE. Kde DELETE vymaže ze serveru daný dokument a TRACE sleduje tok požadavku (např. pokud se daný požadavek vyřizuje přes více distribuovaných proxy serverů).

Dalším důležitým prvkem jsou stavové kódy protokolu. Lze je rozdělit do kategorií dle číselného rozsahu kódů. Kódy s číselným označením od 100 do 199 lze rozlišit jako zprávy informační. Kódy 200 až 299 jsou zprávy oznamující úspěšně provedenou odpověď. Přesměrování je označeno kódy od 300 do 399. Chyby klienta pak kódy 400 až 499. Poslední kategorií stavových kódů jsou chyby na straně serveru, které mají kódové označení v intervalu od 500 do 599. Stavové kódy definované dle RFC jsem převzal z [9].

- 100 Continue - Klient může pokračovat v zasílání požadavku.
- 101 Switching Protocols - Server mění protokol.
- 200 OK - Operace proběhla bez chyby, požadavek je úspěšně splněn.
- 201 Created - Výsledkem požadavku je nově vytvořený objekt.
- 202 Accepted - Byl přijat asynchronní požadavek. Požadavek byl správně akceptován, odpovídající činnost se však ještě zatím nemusela provést.
- 204 No Content - Požadavek byl úspěšný, ale jeho výsledkem nejsou žádná data pro klienta.
- 300 Multiple choises - Požadovaný zdroj se dá získat z několika různých míst. V odpovědi se vrací seznam všech možností.
- 301 Moved Permanently - Požadovaná adresa URL se trvale přesunula na novou adresu URL. Všechny další odkazy musí použít tuto novou URL.

- 302 Moved Temporarily - Požadovaná adresa URL se dočasně přesunula na novou adresu URL. Všechny další odkazy mohou používat dosavadní URL.
- 304 Not Modified - Podmíněný požadavek byl správně zpracován, dokument však od udané doby nebyl modifikován.
- 400 Bad Request Server - nerozumí požadavku, klient jej musí opravit a poslat znovu.
- 401 Unauthorized - Jestliže byl původní požadavek klienta anonymní, musí být nyní autentizován. Pokud už požadavek byl autentizován, pak byl přistup odepřen.
- 403 Forbidden - Server nemůže požadavku vyhovět, autorizace nebyla úspěšná.
- 404 Not Found - Server nenašel zadanou adresu URL.
- 405 Method Not Allowed - Použitá metoda není přípustná pro dosažení požadovaného objektu.
- 406 Not Acceptable - Požadovaný objekt není k dispozici ve formátu podporovaném klientem.
- 408 Request Timeout - Klient nedokončil odesílání požadavku v časovém limitu.
- 410 Gone - Požadovaný objekt byl trvale odstraněn.
- 415 Unsupported Media Type - Požadavek obsahuje data v serveru neznámém formátu.
- 500 Internal Server Error - Na serveru došlo k neočekávané chybě.
- 501 Not Implemented - Tento požadavek server nepodporuje.
- 502 Bad Gateway - Proxy server nebo brána obdržely od dalšího serveru neplatnou odpověď.
- 503 Service Unavailable - Server dočasně nemůže nebo nechce zpracovat požadavek. Většinou když je přetížený nebo se provádí údržba.
- 505 HTTP Version Not Supported - Server nepodporuje verzi HTTP v daném požadavku.

2.4 Jazyky pro popis dokumentu

2.4.1 HTML

HTML (HyperText Markup Language) patří do rodiny značkovacích jazyků SGML (Standart Generalized Markup Language). Jazyk HTML je definován množinou prvků. Každý prvek má příslušný sémantický význam. Jednotlivé prvky mají určitou množinu atributů, které blíže specifikují vlastnosti daného prvku. Dokument je tvořen nadpisy, odstavci, seznamy, obrázky a multimédii. Je definováno také jednoduché propojení dokumentů skrze hypertextové odkazy. HTML dokument se skládá z:

- reference na definici typu dokumentu (DTD) - prvek `doctype`,

- hlavičky dokumentu - prvek `head`,
- těla dokumentu - prvek `body`.

Definice typu dokumentu DTD je kolekce deklarací, která určuje povolenou strukturu prvků a jejich atributů, které se mohou použít v dokumentu [2]. Dále určuje jaká verze HTML byla použita v daném dokumentu. V současné době se využívá DTD HTML 4.01 a XHTML 1.0 Strict. Dle DTD je možné provádět syntaktickou kontrolu dokumentu a zjišťovat, zda daný dokument dodržuje standardy definované W3C konsorciem.

V hlavičce dokumentu je možné uvést název dokumentu - prvek `title` a umožnit propojení se soubory, které jsou nutné pro správnou reprezentaci daného dokumentu. Může se jednat například o reference na soubory s kaskádovými styly nebo skripty jazyka JavaScript. Je nutné také zahrnout prvky typu `meta`, které určují například typ obsahu dokumentu a kódování.

Prvek `body` obsahuje různé další prvky, které formují strukturu dokumentu. Zmíním například prvky jako `div` a `span`, které vymezují blokové a řádkové oblasti v dokumentu. Dále jsou zde prvky jako `table`, prvky pro tvorbu formulářů jako `input`, `select` pro tvorbu rolovacího menu a prvek `p` pro formátování textu do odstavců. Důležitým prvkem je prvek `a`, který definuje hypertextový odkaz s jehož využitím lze jednotlivé dokumenty propojovat a umožnit tak uživateli přístup do jiného dokumentu. Kompletní výčet prvků značkovacího jazyka HTML4 je možné naleznout v [1].

2.4.2 XML

Extensible Markup Language (XML) je značkovacím jazykem pro popis dokumentu. Jeho definice úzce souvisí s definicí jazyka HTML. Společně vychází z univerzálního značkovacího metajazyka SGML (Standard Generalized Markup Language). Jazyk XML je snadno zpracovatelný a využívá se v prostředí internetu. Je vhodný zejména pro výměnu dokumentů v prostředí webu. Prvky jazyka mají podobný formát jako prvky jazyka HTML. V současné době se jeví již jako ne příliš vhodný z důvodu velikosti uvozovacích značek dokumentu. Jako příklad v jazyce XML jsem uvedl záznam skladby 2.1. Z tohoto příkladu je patrné velké množství uvozovacích značek a redundantní informace v nich obsažené. Proto se v dnešní době již často přechází na jazyk JSON (JavaScript Object Notation), který je této redundance zbaven. Tento jazyk bude popsán v podkapitole 2.4.4.

```
<record>
  <active>0</active>
  <artist>Janis Joplin</artist>
  <songname>Cry Baby</songname>
  <songLength>166</songLength>
  <tag>rock</tag>
  <tag>classic rock</tag>
  <tag>hard rock</tag>
</record>
```

Listing 2.1: Příklad záznamu v jazyce XML

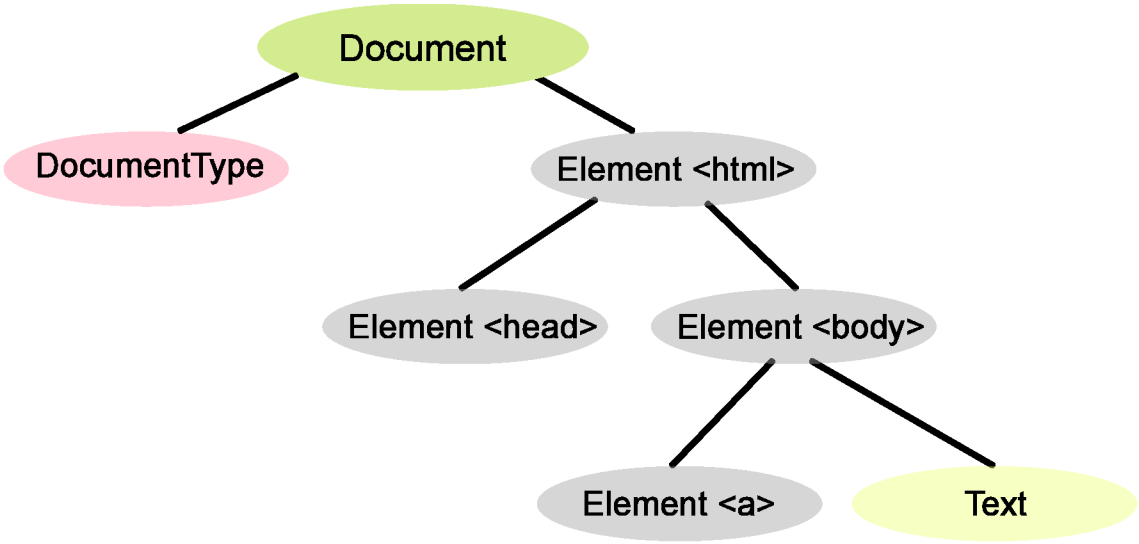
2.4.3 DOM

DOM (Document Object Model) je objektový model dokumentu HTML či XML.

`atributes`	atributy tohoto uzlu
`childNodes`	pole dceřiných uzlů
`documentElement`	kořenový uzel dokumentu
`firstChild`	první dceřiný uzel
`lastChild`	poslední dceřiný uzel
`localName`	lokální název uzlu
`name`	název uzlu včetně jmenného prostoru
`nextSibling`	následující příbuzný uzel
`nodeName`	název uzlu
`nodeType`	typ uzlu
`nodeValue`	hodnota uzlu
`previousSibling`	předchozí příbuzný uzel

Tabulka 2.1: Výčet atributů DOM [13]

Jedná se o jazykově neutrální rozhraní, které umožňuje dynamický přístup programům a skriptům, využívaný například k modifikaci obsahu, struktur a stylů dokumentů [3]. Využívá koncepce objektově orientovaného programování a je platformě nezávislý. Umožňuje do dokumentu přidávat, manipulovat a vyhledávat prvky (elementy). Na DOM lze nahlížet také jako na stromovou strukturu, jejímž kořenovým uzlem je uzel typu Document, který reprezentuje dokument jako celek. Příklad takové stromové struktury je znázorněn na obr. 2.3.



Obrázek 2.3: Příklad stromového znázornění DOM.

Specifikace DOM obsahuje i řadu atributů a metod, které lze využít při práci pomocí rozhraní. Výčet atributů je proveden v tabulce: 2.1.

2.4.4 JSON

JSON (JavaScript Object Notation) [7] je stručný textový formát pro reprezentaci dat odlehčený od zdlouhavých uvozovacích značek, kterých je využito například v HTML nebo XML. Pro přístup není potřeba DOM. Kvůli své stručnosti je vhodný zejména pro výměnu

dat přes HTTP protokol. Jeho definice není spojena s konkrétním programovacím jazykem. Je jednoduše čitelný jak pro člověka tak i pro stroj, který je schopen jej velice snadno a rychle analyzovat, popřípadě generovat. Ve formátu jazyka JSON jsem chopen definovat následující datové typy:

- JSONObject - objekt,
- JSONArray - pole,
- JSONString - textový řetězec,
- JSONNumber - číslo - jak celočíselné, tak reálné s exponentem,
- JSONBoolean - logická hodnota - true, false,
- JSONNull - NULL.

JSON se jeví jako vhodný minimalistický nástupce XML. V současnosti je využíván v řadě API (Application Programming Interface) rozhraní hudebních databází pro vzdálený přístup k datům. Některé aplikace jej využívají jako souborový formát pro ukládání dat. Na příkladu 2.2 je uveden objekt záznamu, který obsahuje řetězce: jméno umělce, název písně. Délku skladby jako celé číslo ve vteřinách a pole hudebních žánrů.

```
{
  record:{
    artist:'Janis Joplin',
    songname:'Cry Baby',
    songlength: 166,
    tags:['rock','classic rock','hard rock']
  }
}
```

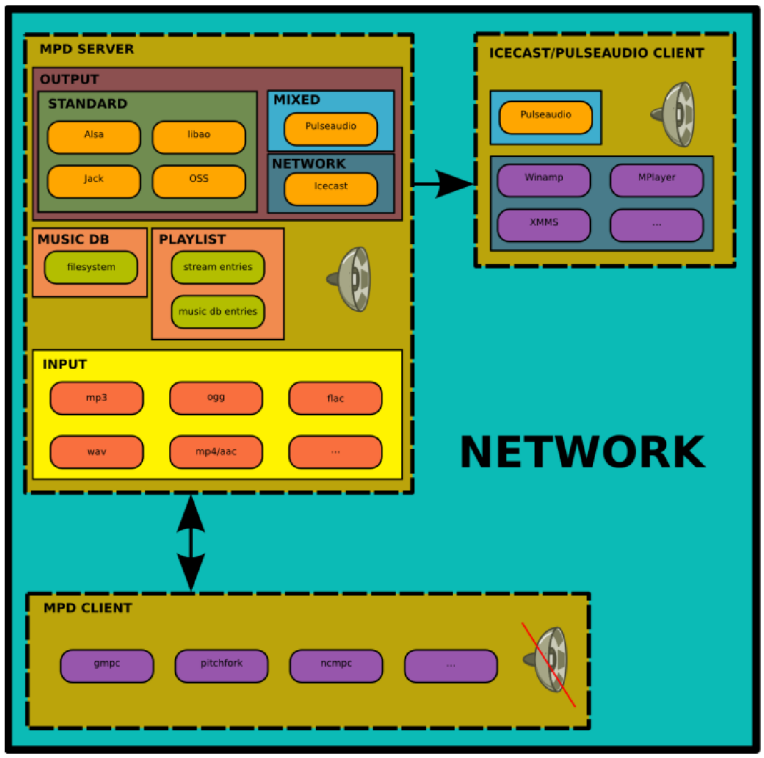
Listing 2.2: Příklad záznamu v jazyce JSON

2.5 Music Player Daemon

Music Player Daemon (MPD) je program serverového typu spouštěný v operačním systému jako démon, tedy služba běžící na pozadí. Slouží jako přehrávač hudby. Hudbu je možné přehrávat jak z lokálního počítače, tak i prostřednictvím streamů z jiných zdrojů. Lze jej tak využít při přehrávání formátů souborů jako: .mp3, .ogg, .wav a dalších. A zároveň jej lze využít jako přehrávač rádií, podcastů a dalších zvukových médií z prostředí Internetu. Kde MPD podporuje protokoly: http, mms, mmsh, mmst, mmsu, gopher, rtp, rtsp, rtmp, rtpt, rtmps.

Pěkným znázorněním struktury MPD serveru je schéma č. 2.4. Schéma je rozděleno do tří hlavních bloků. Prvním blokem je MPD server, který obsahuje vstupní část, ve které umožňuje načítání zvukových souborů. Výstupní část může obsahovat více výstupů. Standardním výstupem je výstup přímo na serveru, kde běží MPD. Ale jelikož se jedná o síťový program, je výstup možné dále šířit pomocí sítě, např. pomocí Icecast nebo Pulseaudio. Tak jak je na schématu naznačeno. A přenášet tak přehrávaný obsah pro více uživatelů v síti.

Součástí MPD serveru je také hudební databáze, která se vytváří ze složky v souborovém systému. Dále umožňuje spravování playlistů. Na MPD server je samozřejmě napojena klientská aplikace, která jej přes síťové rozhraní umožňuje řídit.



Obrázek 2.4: MPD model

MPD server poskytuje uživateli síťové rozhraní, přes které je možné ovládat základní i rozšířené funkce přehrávače. Nyní budu popisovat konfiguraci na systému Linux. Dle standardně definovaného konfiguračního souboru `mpd.conf` umístěného v adresáři `/etc` je rozhraní spuštěné na portu 6600 a využívá protokol TCP. Dalším důležitým parametrem konfigurace je určení složky, ve které se budou vyhledávat jednotlivé skladby a nad kterou vytvoří MPD hudební databázi. Konfigurace složky se provádí volbou `music_directory` v konfiguračním souboru.

Uživatel pak přistupuje k jeho funkcím serveru prostřednictvím klienta, který zasílá jednoduché příkazy v kódování UTF-8. Příkazy jsem dle specifikace protokolu[6] rozdělil do šesti kategorií na:

- Administrační příkazy,
- Informační příkazy,
- Databázové příkazy,
- Playlistové příkazy,
- Playbackové příkazy,
- Ostatní příkazy.

Mezi administrační příkazy patří například příkazy pro zapnutí a vypnutí zvukového výstupu (`enableoutput`, `disableoutput`), korektní zastavení MPD serveru (`kill`), aktualizaci hudební databáze (`update`).

Další kategorií příkazů jsou informační příkazy. Tato kategorie zahrnuje např. příkaz k zjištění aktuálního stavu MPD serveru (`status`). Stav je charakterizován úrovní hlasitosti, stavem volby pro opakování přehrávaných skladeb, stavem volby pro náhodný výběr skladby, následující skladbou apod. Dalšími příkazy spadající do kategorie informačních příkazů jsou například příkazy pro zobrazení všech výstupů (`outputs`), pro zjištění dostupných příkazů (`commands`). Jedním z důležitých příkazů je příkaz (`tagtypes`), který vrací seznam typů metadat přístupných k dané skladbě.

Jelikož MPD obsahuje také vlastní databázi skladeb, jsou nutné databázové příkazy, které umožňují např. prohledávání (`search,find`), výpis všech adresářů a souborů v databázi (`listall`).

Nejrozsáhlejší kategorií jsou příkazy pro práci s playlistem. Tato kategorie zahrnuje příkazy pro přidání, odebrání skladby (`add,delete`), přesun skladby v rámci playlistu (`move`), vymazání playlistu (`clear`), zobrazení metadat k právě přehrávané skladbě (`currentsong`). Jelikož MPD spravuje kompletní seznamy playlistů, další příkazy se vztahují právě k této funkci. Jedná se například o příkazy pro přidání (`playlistadd`), načtení (`load`), přesunutí (`playlistmove`), přejmenování (`rename`), vymazání (`playlistclear`) a odstranění (`playlistdelete`) playlistu.

Předposlední kategorií jsou příkazy pro ovládání funkcí přehrávače. Jedná se o klasické funkce jako je spuštění přehrávání (`playid`), pozastavení přehrávání (`pause`), zastavení přehrávání (`stop`), nastavení úrovně hlasitosti (`setvol`), zapnutí náhodné volby pro výběr skladby v playlistu (`random`), zapnutí opakování (`repeat`), seekování (`seek`), přepnutí na další skladbu (`next`) a další.

Poslední kategorií jsou ostatní příkazy, které nejsou již dále kategorizované. Zahrnují např. příkaz pro ukončení spojení se serverem (`close`), autentizační příkaz pro zadání hesla (`password`), příkaz pro testování odezvy serveru (`ping`).

Kapitola 3

Rozbor současných klientských aplikací

V současné době jsou na internetu k dispozici stovky klientských aplikací, které umožňují kompletní ovládání MPD serverové aplikace. Klienty je možné dělit dle různých kritérií.

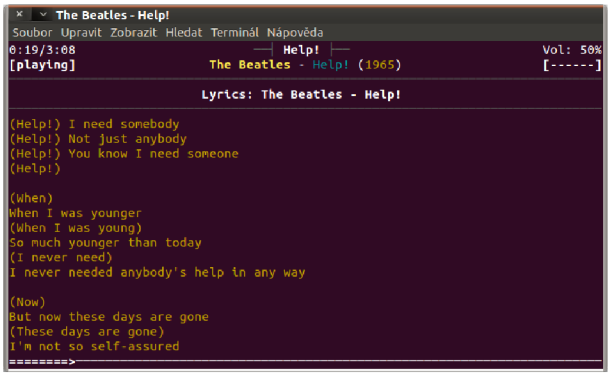
Prvním kritériem by mohlo být rozdělení dle programovacích jazyků. Existují klienti napsaní v klasických programovacích jazycích jako je např. jazyk C, C++, C#, Java. Dále pak v dynamických skriptovacích jazycích jako jsou Perl, Python, Ruby, PHP a dalších. Je možné je provozovat na různých platformách od operačního systému Windows, Linux až po MacOS, případně je lze spouštět a provozovat přímo v internetových prohlížečích (např. pitchfork, patchfork). Nyní se také rozšiřují klienti pro mobilní platformy jako je Google Android, Apple IPhone, Apple IPad či Microsoft Windows Mobile/PocketPC. Dalším členěním by mohlo být členění rozlišující typ grafického uživatelského rozhraní, kde se od klientů běžících v příkazové řádce dostáváme až po aplikace napsané různých knihovnách a toolkitech jako jsou např. GTK+ či Qt.

Jelikož je implementace MPD protokolu velmi jednoduše provedena za pomocí TCP/IP spojení, je implementace klientských aplikací, které umí pouze ovládat tento server, triviální záležitostí. Proto je mnoho z klientských aplikací velmi jednoduchých a pro rozbor a využití v této práci nevyhovujících. Vybral jsem proto několik klientských aplikací, které nejen že dovedou ovládat MPD server, ale zobrazují také další užitečné informace (metadata) o přehrávané skladbě. Příkladem takových informací může být např. informace o interpretovi, informace o přehrávané skladbě, obaly CD, texty k skladbám, podobné skladby, žánry, informace z Wikipedie, kytarové akordy dané skladby apod.

3.1 NCMPCPP

Jediným textovým klientem, který se ve většině vlastnostech dokáže vyrovnat vlastnostem grafického klienta je NCMPCPP [4] (NCurses MPd Client) nebo také NCMPC++. Komplikovaný název je tvořen z názvu knihovny Ncurses a názvu programovacího jazyka C++, které klient využívá. Jde o projekt vycházející z projektu NCMPC doplněný však o reprezentaci relevantních metadat. V aplikaci je možné zobrazit metadata jako jsou texty písní (klávesa l), informace o interpretovi (klávesa I) a informace o skladbě (klávesa i). Data pro hledání interpreta a písně se využívají z metadat u jednotlivých souborů dle kontejnerů ID3, Vorbis apod. Vzhledem ke konzolové orientaci programu není možné zobrazovat obrazové či audiovizuální materiály.

Užitečnou vlastností je možnost vybírání zdroje pro získávání textů písní. Po stisknutí klávesové kombinace `SHIFT+L` se vybere jiná databáze. Ve verzi 0.5.6-2 je možné vybírat z devíti databází s dodatečnou volbou hledání ve všech databázích.



Obrázek 3.1: Ukázka NCMPCPP klientské aplikace.

3.2 Sonata

Další aplikací, která je již od klienta NCMPC uživatelsky příjemnější, je aplikace Sonata. Sonata je nyní příkladem aplikace, ve které byla využita vhodná myšlenka reprezentace metadat. V současné verzi 1.6.2.1, která je dodávána do linuxové distribuce Ubuntu, ale bohužel většina reprezentací metadat nefunguje. Nefunguje stahování obalů ze serverů Amazon, jelikož došlo ke změně přístupu k Amazon Web Services API rozhraní a nyní je již přístupné pouze po přihlášení a vygenerování unikátního Access Key ID a Access Key. Nespolehlivé je také získávání textů z LiricsWiki, které funguje jen u části interpretů.

Obrázek 3.2: Ukázka z klientské aplikace Sonata.

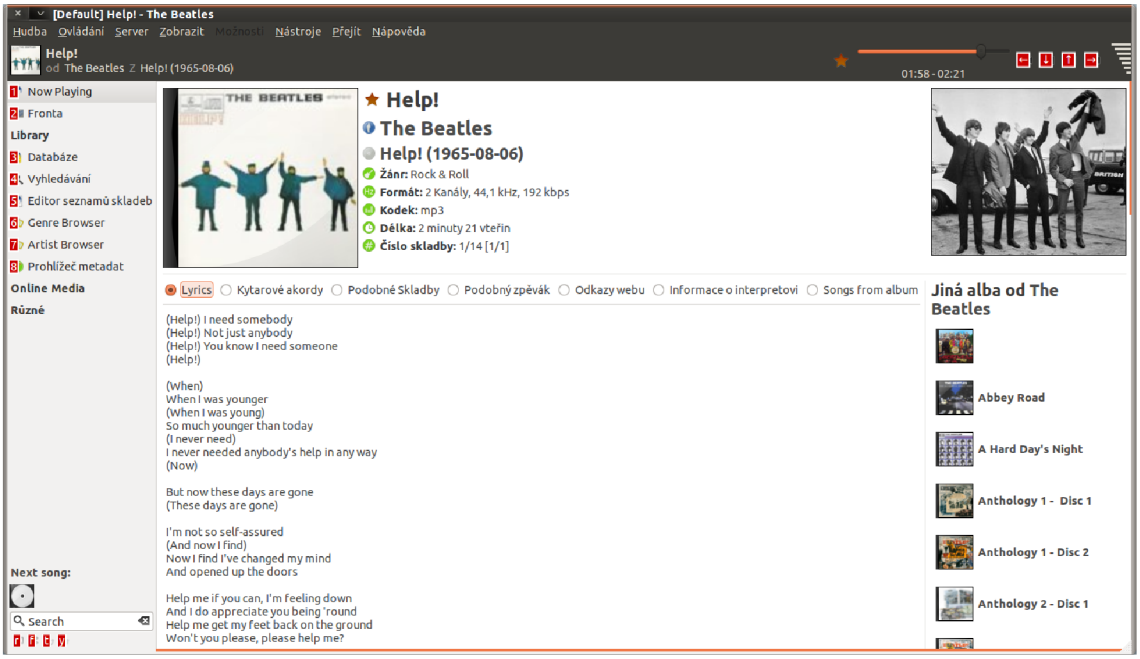
Výhodou aplikace je možnost ukládat vyhledávané obaly alb a texty. Pro tyto účely jsou vytvořeny dvě složky přímo v domovském adresáři uživatele. Jedná se o složky `.covers` a `.lyrics`. Obaly se uchovávají jako obrázky JPEG s názvem ve tvaru `[Interpret]-[Album].jpg`.

Sonata kromě složky `.covers` prohledává i jiné složky v lokálním souborovém systému, kde vyhledává s pomocí filtrace názvů souborů. Texty k písním jsou uchovávány v textových souborech s koncovkou `.txt`.

3.3 Gnome Music Player Client

Nejsofistikovanější ze všech klientských aplikací z hlediska reprezentace metadat k přehrávané skladbě je Gnome Music Player Client. Jedná se o multiplatformní klientskou aplikaci s možností rozšíření za pomoci pluginů. Aplikace interaktivně sleduje přehrávané skladby a k přehrávané skladbě dodává například: obal alba, fotografie interpretů, texty, kytarové akordy. Zároveň doporučuje podobné skladby, ať už nalezené v knihovně skladeb na pevném disku, nebo cizí interprety, kteří se v knihovně ještě nenachází. Dále doporučuje jiná alba od interpretů a zobrazuje odkazy do hudebních databází na internetu.

Zdrojem metadat jsou z velké části data z hudebních databází, která jsou relevantním zdrojem informací a zaručují do jisté míry správnost. Zdroje informací však nelze měnit a jsou pevně definované. Jestliže se zdrojová adresa jakkoliv změní, například úpravou tvaru URL adresy na webu, informace jsou pak často nedostupné a je nutné vyčkat na úpravu od samotných autorů aplikace, což může trvat i několik dní.

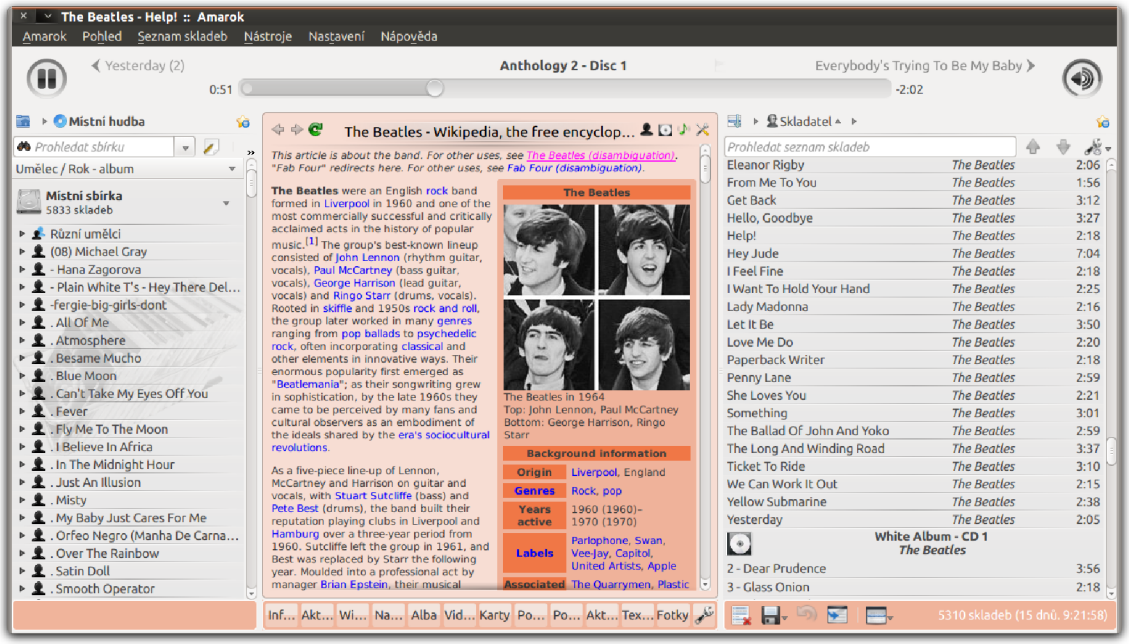


Obrázek 3.3: Ukázka z klientské aplikace Gnome Music Player Client.

3.4 Amarok

Poslední aplikací, kterou zmíním je aplikace Amarok. Narozdíl od předešlých aplikací se nejedná o klientskou aplikaci nevyužívající MPD server ale o samostatný hudební přehrávač. Z hlediska úrovně a prezentace metadat je ale aplikace zajímavá. Na obr č.3.4 je zobrazena prezentace metadat přímo ze stránek anglické verze encyklopedie Wikipedia. Zajímavý je

způsob prezentace, kdy se zobrazuje kompletní webová stránka namísto pouhých vybraných informací. Z databáze Last.fm je v další záložce možnost procházet autory skupiny, podobné umělce, texty skladeb. Z uživatelské databáze flickr.com se zobrazují fotografie nahrané uživateli této služby. Bohužel někdy jsou fotografie chybně označeny a tak dochází k zobrazování nesouvisejících fotografií. Zajímavá je také prezentace kytarových akordů. Po hlubším prozkoumání jsem zjistil, že zdrojem metadat, jako u předešlých klientských aplikací, jsou také API rozhraní do hudebních databází. Hudební přehrávač tedy zašle informace o aktuální přehrávané skladbě přímo na vybraný server a v odpovědi jsou vráceny nalezené a relevantní výsledky. Inteligence pracování s daty je závislá na datech poskytovaných třetí stranou.

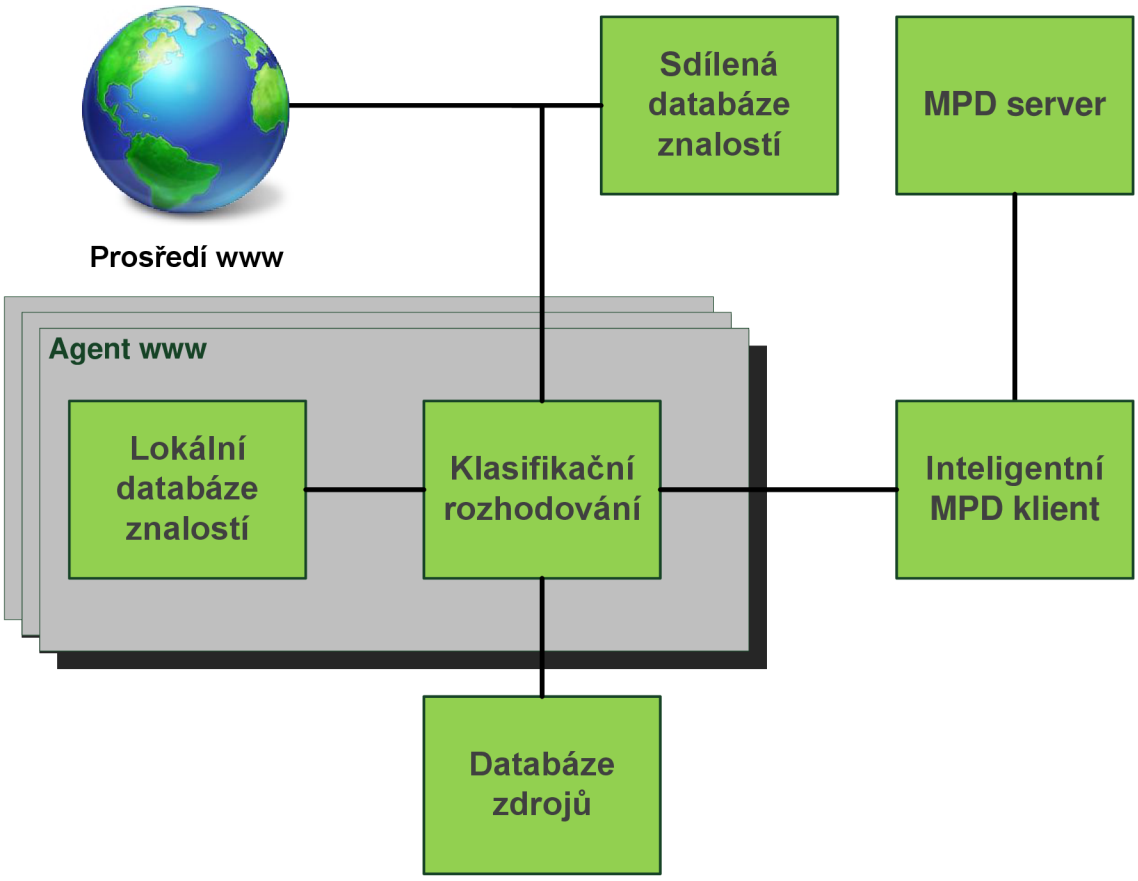


Obrázek 3.4: Ukázka z hudebního přehrávače Amarok.

Kapitola 4

Návrh implementace

V této kapitole se zaměřím na návrh implementace a využití inteligentního klienta. Na obrázku č. 4.1 je uvedeno schéma návrhu aplikace.



Obrázek 4.1: Schéma inteligentního klienta

Schéma se skládá z několika modulů. Jak je naznačeno na modelu, návrh předpokládá multiagentní prostředí. Kdy každý agent spravuje informace z jednoho informačního zdroje, který načítá z lokální databáze zdrojů. Agenti poté prochází zdroj a naplňují lokální databázi znalostí. Ta je tvořena informacemi, které se vyhledávají v daném zdroji. V případě nedostatku informací se může obrátit na ostatní agenty. Spolupráce na řešení problému je

při hledání informací důležitá, neboť při následné klasifikaci dat je úspěšnost klasifikace úplného záznamu vyšší. Modul klasifikačního rozhodování pak umožňuje posouzení relevance nalezených informací k přehrávanému obsahu. Dle teoretického rozboru by klasifikační rozhodování mohlo být implementované s využitím bayesovské klasifikace dat. Zde je patrná souvislost ve využití filtrování nevyžádaných emailových zpráv. Bayesovská klasifikace také umožňuje klasifikaci neúplných záznamů při velkém množství trénovacích záznamů, což je ve webovém prostředí jistě výhodou. Dalším modulem je modul inteligentního klienta. Tento modul ve své podstatě shromažďuje data od agentů a prezentuje je v grafickém uživatelském rozhraní (GUI). Uživateli pak prostřednictvím GUI umožňuje vyjádřit nesouhlas se zobrazovanou informací. Agentům zároveň z MPD serveru poskytuje informace o právě přehrávaném obsahu. Sdílená databáze znalostí zahrnuje trénovací záznamy z lokální databáze znalostí shromážděných od více inteligentních klientů.

4.1 Zdroje informací

Pro inteligentního klienta nezbytná součást. Jako zdroj je v rámci této práce chápána webová stránka v jazyce HTML, XML či JSON. Ze zdrojů inteligentní klient čerpá podstatné informace, dokáže se v nich orientovat a klasifikovat je do tříd. Případně rozhodovat, zda je informace důležitá, související apod. Zdoje informací jsem rozdělil na:

1. Primární - ověřené zdroje informací - většinou databáze,
2. Sekundární - informační weby, menší žánrové weby, fotogalerie, video archivy.

Jako primární zdroj lze chápat ověřený zdroj, kterým mohou být např. hudební databáze. Těch existuje na internetu velké množství. Jako příklad uvádím databáze last.fm, musicbrainz.org. Vyjmenované databáze jsou databáze pro všechny hudební žánry. Primární zdroje by měly sloužit jako podpůrné zdroje pro učení inteligentního klienta v případech, kdy nebude k dispozici žádná databáze znalostí.

Sekundární zdroj je chápán ve významu menších žánrových webů a databází. Většinou se jedná již o žánrově separované weby. Z těchto zdrojů mohou být získávány také cenné informace mnohdy informačně hodnotnější či doplňující zdroje primární. Sekundární zdroje by mělo být možné přidávat do inteligentního klienta a ten by měl být schopen se v nich orientovat a dle žánrového zaměření v nich také vyhledávat.

Informace vyhledávané a interpretované inteligentním klientem lze rozdělit do třech kategorií:

1. textové,
2. grafické,
3. audiovizuální.

Do textových dat lze zařadit např. informace o zpěvákovi nebo kapele, informace o skladbách například: texty, akordy, data složení, žánrové informace, názvy alb, hodnocení. Z těchto dat pak lze vyvodit a rozčlenit např. informace o podobných interpretech či skladbách. Grafická data představují obrázky interpreta, kapely, fotografie z koncertů apod. Audiovizuální data jsou např. videoklipy písní, písní z koncertů, dokumentární videoklipy apod.

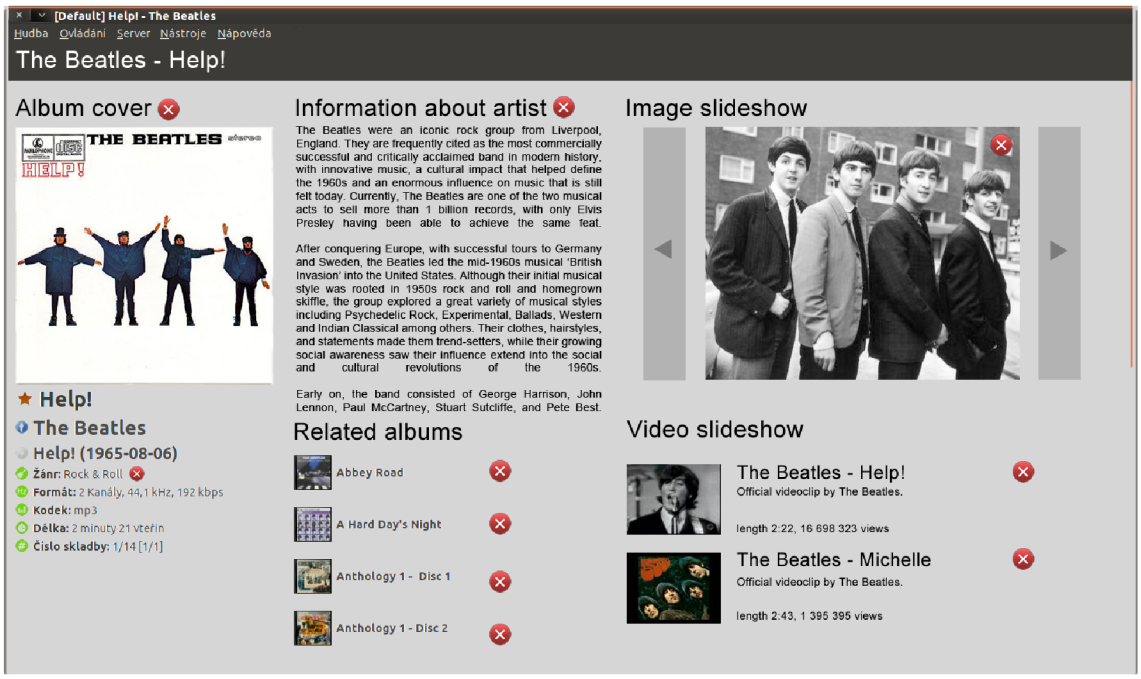
4.2 Zjišťování informací

Jedním z hlavních problémů je získávání informací do databáze znalostí, podle kterých se budou informace klasifikovat. V prostředí jazyka HTML se dle knihy Mining The Web [5, s.188] podstatné informace dokumentu nachází v nadpisech (prvky `title, h1-h6`), části v textu vyznačené tučně či kurzívou (prvky `strong, em`), dále pak v seznamech (prvky `ul, ol, li, df, dt`), tabulkách a dalších prvcích. Dokumenty jsou samozřejmě propojené hypertextovými odkazy. Ty lze využít také ke směrování k relevantnějšímu obsahu. Hypertextové odkazy (prvky `a`) jsou uvozené na textovém řetězci, který vypovídá o informacích, vyskytujících se na odkazovaném dokumentu. Nedílnou součástí při procházení odkazů je sledování URL adresy. V adrese se také mohou nacházet další podstatné informace.

Jedním z návrhů, kterým bych rád rozšířil a zrychlil zjišťování informací z daného webu, by měla být implementace inteligentního vyhledávání. Kdy při prvotním procházení webové stránky bude detekován formulář, který umožní vyhledávat informace přímo na webu bez nutnosti procházení cesty hypertextových odkazů. Detekce vyhledávacího formuláře by se měla vyskytovat na úrovni jazyka HTML. Principiálně lze popsat detekci jako zjištění výskytů prvků `form` a uvnitř prvků `input`. Kdy z prvku `form` lze dle jeho atributů určit HTTP metodu odesílání a cílovou URL adresu.

4.3 Návrh GUI

Grafické uživatelské rozhraní je přizpůsobeno funkčnosti, kdy je hlavní důraz kladen na interpretaci metadat a označení chybné nebo nesouvisející informace. Plocha uživatelského rozhraní aplikace bude rozdělena na menší podoblasti. Do podoblastí bude možné zvolit druh metadat. Na obr.4.2 je provedena vizualizace uživatelského rozhraní aplikace. Vyjádření chyby v zobrazovaných metadatech se může u jednotlivých druhů metadat lišit.



Obrázek 4.2: Návrh podoby grafického uživatelského rozhraní.

Vyjádření nesouhlasu s prezentovanými metadaty je možné provést kliknutím na červenou ikonu s křížkem. Dle typu metadat by na tuto akci mělo být zobrazeno příslušné dialogové okno, kde bude formulářovým typem doplněna míra nesouhlasu s touto informací (např. výběr hodnocení od 1 do 5).

4.4 Návrh databáze znalostí

Pro implementaci lokální databáze znalostí bude využito lokálního souborového systému. Návrh předpokládá, že každý záznam z databáze znalostí bude reprezentován souborem ve formátu JSON notace. Především kvůli rychlosti zpracování, snadné interpretovatelnosti a úspoře místa. Tento počáteční návrh je vhodný také pro sdílení znalostí prostřednictvím sdílené databáze znalostí.

Sdílená databáze znalostí obsahuje veškeré získané znalosti od všech inteligentních klientských aplikací. Oproti lokální databázi je především objemnější databází. Přístup do sdílené databáze znalostí je klientovi umožněn prostřednictvím HTTP protokolu. Dotazem získá záznamy ve formátu JSON. Zasílání lokálních znalostí do sdílené databáze probíhá prostřednictvím metody POST protokolu HTTP. Sdílenou databázi znalostí tak teoreticky mohou využívat i jiní inteligentní klienti, kteří mohou být rozšířením či specializací inteligentní klientské aplikace vytvořené v této práci.

Kapitola 5

Implementace klientské části

Klientská část a její implementace byla především z důvodu přenositelnosti a potřeb specifických knihoven provedena kompletně v jazyce C++. Pro grafické uživatelské rozhraní (GUI) jsem využil nejnovějších knihoven GTK 3 implementovaných v rozhraní GTKMM pro C++. Program je napsaný tak, aby byl přenositelný na většinu podporovaných operačních systémů. V implementaci bylo využito těchto knihoven:

- GTKMM 3 - libgtkmm-3.0-dev,
- CURL - libcurl4-openssl-dev,
- XML - libxml2-utils,
- XSLT - libxslt-dev,
- JSONCPP - libjsoncpp-dev,
- MPDCLIENT - libmpdclient-dev,
- INTLTOOL - intltool.

Pro komunikaci prostřednictvím protokolu HTTP a HTTPS byla využita knihovna Curl, která zapouzdřuje komunikaci mezi klientem a webovými servery. K parsování webových stránek je využito knihovny libxml, která ve své části obsahuje HTML parser pro standard HTML Transitional v.4. Pro potřeby dodatečných operací nad HTML dokumenty je ještě doplněna o knihovnu XSLT. Důležitou vlastností je možnost paralelního zpracování více webových stránek současně. Standard HTML Transitional v.4 je v současné době již zastaralý. Bohužel jsem nenalezl knihovnu pro jazyk C++, která by umožňovala zpracovat nový standard HTML5. K ukládání nalezených informací do formátu JSON je do programu začleněna knihovna jsoncpp. Komunikaci mezi klientem a MPD serverem zajišťuje knihovna MPD client. Pro potřeby lokalizace uživatelského rozhraní je použita knihovna intltool. Jako kompilátor byl použit kompilátor jazyka C++ ve verzi 4.6. Pro zjednodušení detekce knihoven v uživatelském operačním systému je pro celý projekt použit nástroj automake. Kompilace a potřebné knihovny jsou tak tímto nástrojem automaticky zjišťovány a bez jejich přítomností není možné program kompilovat do binárního kódu.

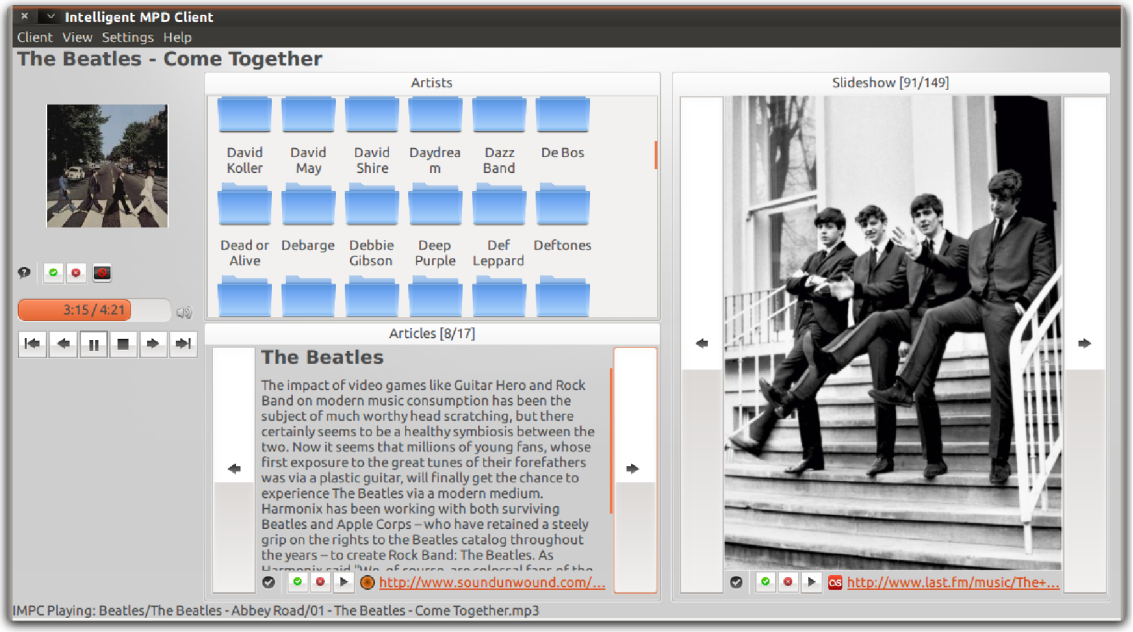
V oblasti klasifikace dat nebyla využita externí knihovna. Pro použití bayesovské klasifikace jsem zvažoval využití knihoven: Bayes++[1], naive-bayes-classifier[2] a nBayes[3]. Po pro-

[1] http://bayesclasses.sourceforge.net
[2] http://code.google.com/p/naive-bayes-classifier/
[3] http://nbayes.codeplex.com

studování dokumentace knihovny Bayes++ jsem usoudil, že úprava pro potřeby inteligentního klienta by byla příliš velká a neefektivní. Knihovna je velmi detailní a rozsáhlá. Obsahuje i další klasifikační modely, které nejsou v případě implementace nutné. Další prostudovanou knihovnou je naive-bayes-classifier. Testování knihovny prokázalo nesrovnalosti při klasifikaci dat a ve výsledcích. Poslední knihovnou je knihovna nBayes. Po prostudování zdrojových kódů se knihovna prokázala jako nejvhodnější pro použití. Bohužel nebyla implementována pro jazyk C++. Knihovna je uveřejněna pod licencí Mozilla Public License. Proto jsem důležité části této knihovny přepsal do jazyka C++ s referencemi na originální zdrojové kódy.

5.1 Uživatelské rozhraní

Tvorba uživatelského rozhraní proběhla kompletně dle návrhu. Implementace struktury uživatelského rozhraní proběhla v aplikaci Glade pro GTK. Uživatelské rozhraní je primárně koncipováno pro možnost prezence nalezených relevantních dat. Na obr.5.1 je viditelné hlavní okno aplikace. Je rozděleno na několik částí. Uživatelské rozhraní je rozděleno do komponent(widgetů). Levý bok obsahuje komponentu zobrazující obaly alb k aktuálně přehrávané skladbě. Pod obalem se nachází klasické ovládací prvky hudebních přehrávačů jako je posunování skladby, změna hlasitosti a další ovládací tlačítka přehrávání. Vpravo je již plocha pro zobrazení dalších widgetů. Na obrázku je vidět widget s přehledem autorů (Artist Widget), kteří jsou v databázi MPD serveru. Volit aktuálně přehrávanou píseň je možné právě přes tento widget. Rozkliknutím konkrétního autora se zobrazí jeho interpretované skladby. Pod widgetem pro výběr skladby se nachází komponenta pro zobrazování nalezených článků a textu (Articles Widget). Úplně napravo se nachází komponenta pro zobrazení všech souvisejících obrázků a fotografií. Jednotlivé widgety jsou přesouvatelné. Přesuny lze realizovat zatím pouze mezi záložkovacími widgety, které jsou zobrazené na obr.5.1.

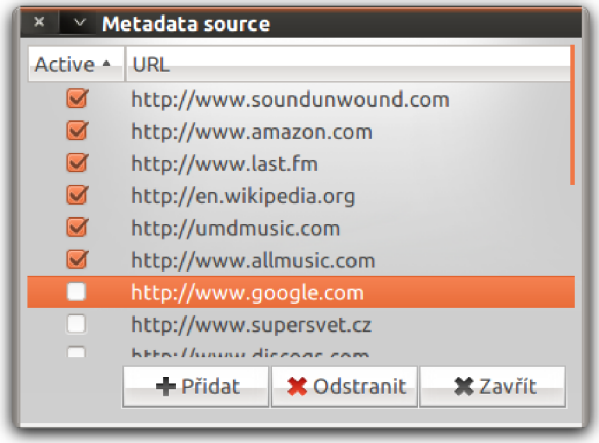


Obrázek 5.1: Podoba implementovaného uživatelského rozhraní

Vyjádření zpětné vazby uživatele k prezentovanému obsahu lze nalézt pod každým widgetem. Každý widget obsahuje lištu tak jako na obr.5.2. Na této liště je zleva zobrazena ikonka, která určuje třídu (správnost) právě zobrazovaného objektu ve widgetu. Lze rozlišovat tři stavy této ikony: správný obsah (viz. obr.), špatný obsah (křížek) a neklasifikovatelný obsah (otazník). Dále následují dvě tlačítka pro možnost ovlivnit třídu zařazení daného objektu. Každý uživatel se může k danému objektu vyjádřit pouze jednou. Ikonka se šipkou umožňuje spuštění nebo pozastavení prezentace objektů ve widgetu. Na konec následuje adresa zdroje, ve kterém byl objekt nalezen.

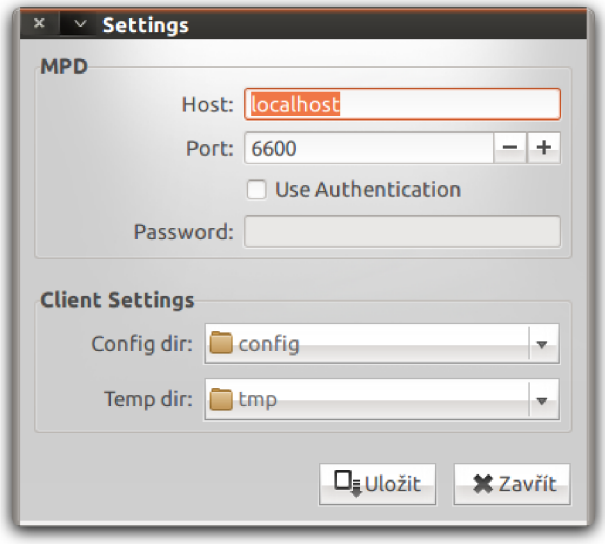
Obrázek 5.2: Lišta pro zobrazení informací o zdroji s možností zpětné vazby.

K nastavení potřebných parametrů a vstupů programu jsou k dispozici dvě dialogová okna. První okno (obr.5.3) je okno pro nastavení zdrojů dat. V okně lze jednoduše přidávat a odebírat zdroje, ze kterých pak klientská aplikace zjišťuje potřebná data. V okně lze uchovávat také neaktivní zdroje. Druhým oknem je dialog nastavení (obr.5.4). V tomto okně lze specifikovat údaje o MPD serveru a o adresáři, ve kterém se budou uchovávat dočasně stažené soubory.



Obrázek 5.3: Dialogové okno pro nastavení zdrojů dat.

Možnosti rozšíření uživatelského rozhraní jsou široké. Od tvorby nových, specifičtějších komponent, až po změnu stávajících. Vhodným prvkem by byla možnost rozšířit jednotlivé widgety o možnosti změny jejich velikosti. U některých typů widgetů to může být užitečné. U jiných z hlediska variability zobrazovaných dat nikoliv. Jedním z plánovaných rozšíření widgetů by mohlo být rozšíření Artist Widgetu. Kde by se místo ikon složek zobrazovaly přímo portréty autorů. Z hlediska návrhu aplikace by toto rozšíření bylo triviální. Různorodý obsah fotografií by ale musel být ještě dále zpracován např. algoritmy pro detekci obličeje, postavy apod. Fotografie by pak mohly být ořezány a umístěny místo ikon. Dalším, zatím nerealizovaným rozšířením by se nabízela možnost zobrazování videí či změn pozadí dle aktuálně přehrávaného obsahu.



Obrázek 5.4: Dialogové okno nastavení aplikace.

5.2 Implementovaná funkčnost

5.2.1 Konfigurace

Aplikaci je možné konfigurovat prostřednictvím dvou konfiguračních souborů nebo prostřednictvím grafického uživatelského rozhraní. Soubory se nachází v adresáři `config`. První z `settings.json` obsahuje konfigurační nastavení pro připojení k serveru MPD, přepínače pro možnosti sdílení informací se vzdáleným datovým skladem a konfiguraci složky, do které se budou ukládat dočasné soubory. Druhým souborem je soubor pro volbu zdrojů `sources.json`. Příklad konfiguračního souboru jsem uvedl v zdrojovém kódu č.5.1. Soubor pro volbu zdrojů byl implementován v jazyce JSON, kvůli možnostem snadného sdílení mezi uživateli.

5.2.2 Prezentace dat

Hlavní požadavek kladený na klientskou aplikaci byl, aby poskytovala co největší možný prostor pro prezentaci nalezených dat. Tento požadavek byl splněn a aplikace poskytuje možnost umístění komponent pro prezentaci. Byly implementovány prozatím dvě komponenty, které provádějí zobrazování dat. První komponentou je komponenta pro zobrazování textových článků, druhá zobrazuje obrazové informace. Hlavní struktura komponent se ve své podstatě a možnostech neliší.

5.2.3 Hodnocení dat a klasifikace

Každý uživatel inteligentního klienta má možnost vyjádřit souhlas či nesouhlas s prezentovanou informací. Každý prezentovaný záznam může být hodnocen uživatelem. Klient si uživatelské hodnocení ukládá a hodnotit informaci podruhé již zakazuje. Pokud v hodnocení převažuje větší míra souhlasu s vybranou informací, je předána klasifikátoru jako informace se správným obsahem. Pokud převažuje záporné hodnocení, je informace předána klasifikátoru jako nesprávná. V případě shodného počtu hlasů se klasifikátor z informace neučí.

```
{
   "sources" : [
      {
         "active" : false,
         "url" : "http://www.google.com"
      },
      {
         "active" : false,
         "url" : "http://www.discogs.com"
      },
      {
         "active" : true,
         "url" : "http://www.last.fm"
      },
      {
         "active" : false,
         "url" : "http://umdmusic.com"
      }
      {
         "active" : false,
         "url" : "http://en.wikipedia.org"
      },
      {
         "active" : true,
         "url" : "http://www.seznam.cz"
      }
   ]
}
```

Listing 5.1: Příklad konfiguračního souboru sources.json v jazyce JSON

5.2.4 Sdílení

Součástí aplikace je také možnost sdílení a synchronizace dat s referenčním sdíleným datovým skladem. Sdílení dat aplikace řeší bez zásahu uživatele. Uživatel může pouze volbou přepínače v menu nastavení aplikace sdílení zapnout či vypnout. Informace se předávají vzdálenému webovému serveru, který je prozatím pevně stanovený. Stažení sdílených informací ze vzdáleného datového skladu probíhá při spuštění přehrávaní konkrétní skladby. Odeslání informací se uskuteční po skončení přehrávané skladby.

5.3 Zpracování dat

5.3.1 Průchod

Průchody weby jsem rozdělil na dva případy. V prvním případě se na webu nachází vyhledávací formulář, který inteligentní klient dokáže využít a skrze něj se tak rychleji dostat k hledanému obsahu. Weby obsahující vyhledávací pole jsou například rozsáhlé databáze autorů a jejich děl, databáze obrázků (Picasa, Flickr) či fulltextové vyhledávače (Google, Seznam apod.).

Implementace detekce vyhledávacího pole spočívá v nalezení příslušných prvků typu `input`, které je možné odeslat skrze prvek typu `form`. Inteligentní klient je schopen využít

jak metodu `GET`, tak metodu `POST` HTTP protokolu. Na stránce se mohou vyskytovat i jiné druhy vstupních prvků než jen vyhledávání. Může se jednat se například o přihlašování, třídění, podrobné vyhledávání apod. Formuláře těchto druhů jsou detekovány a v průchodu přeskočeny.

Druhým případem jsou stránky, které neobsahují vyhledávání, často konkrétní, specifické weby zaměřené na daného autora. Tyto weby je nutné procházet dle vstupních parametrů (název autora, skladby nebo alba) skrze hypertextové odkazy. V rámci řešení jsem také uvažoval multijazyčnost programu. Proto nelze v průchodu hledat klíčová slova, která v jiných jazycích neexistují či mají jinou syntaktickou podobu. V těchto případech by bylo možné využít překladačů. Rekurzivní sestup jsem v rámci řešení neuvažoval, neboť by se klient při zpracování více zdrojů mohl zahltit nerelevantními informacemi (v případě, že není ještě dostatečně naučen) nebo by mohl být klientské aplikaci zakázán přístup kvůli nadměrnému zasílání požadavků na server.

Jednotlivé průchody weby, které jsou prováděny pro každý zdroj, jsou v rámci řešení implementace paralelizovány. Každý průchod si současně udržuje historii navštívených a zpracovaných webových adres aby nemohlo dojít k zacyklení. Mimo stanovených základních vyhledávačů (zatím Google, Seznam, Yahoo) se průchod webem omezuje na doménu 2.řádu vybraného zdroje.

5.3.2 Detekce textu

Jednou z klíčových vlastností úspěšně fungujícího inteligentního klienta je detekce textu. Současné prostředí internetu je nesémantické. O sémantickém webu se v souvislosti s cíleným a konkrétním vyhledávání informací diskutuje již řadu let. Byly navrženy i standardy Resource Description Framework (RDF) a Ontology Web Language (OWL). Sémantický web by rozšířil možnosti dnešního historického modelu webu, který je tvořen dokumenty, propojenými hypertextovými odkazy. Současný model přetrvává, jelikož přejít na sémantický web je možné pouze změnou standardů. Částečným přiblížením k sémantickému webu je standard HTML5. Z hlediska implementace jsem jej bohužel nemohl uvažovat, jelikož jsem nenalezl vhodnou knihovnu pro zpracování.

V současném webovém prostředí je detekce textových informací náročná. Z vizuálního pohledu uživatele na web se to nemusí zdát jako problém. Ale jelikož byly webové dokumenty na internetu doplněny o možnosti definic vizuálních stylů (CSS), prostřednictvím kterých je možné měnit pořadí zobrazených struktur webů, je detekce značně ztížena a výsledek prohledávání tak při strojovém zpracování nemusí být pokaždé srozumitelný.

Při implementaci detekce textů jsem zvažoval tyto problémy. Jelikož prezentované informace inteligentním klientem nemusí mít podobu kompletní webové stránky, tak jsem informace v rámci jedné webové stránky rozdělil. Dělení jsem uskutečnil dle nadpisů (prvky typu `H1-H6`), kde se při detekci jednotlivých odstavců v rámci struktury dokumentu hledá nejbližší relevantní nadpis. Dvojice nadpis-odstavec je pak ukládána. Tuto možnost osobně hodnotím jako možnost s nejmenší možnou pravděpodobností chyby při zachování jednoduchého zpracování. Prezentace kratších a zajímavějších dat je také dle mého názoru lépe vyhovující než prezentace příliš dlouhých textů. Dvojice pak může být dále klasifikována, přičemž se tak oddělí související a nesouvisející obsah.

5.3.3 Detekce obrázků

Obrázky je možné z prostředí webových dokumentů získávat jednoduchým rozparsováním dokumentu. Dle standardů definovaných konsorciem W3C je povinným atributem prvku

typu `img` vlastnost `alt`. Zde by měl být popis daného obrázku. Další z vlastností může být vlastnost `title`, která je však nepovinná. Mnoho webových stránek ale tyto vlastnosti ignorují a nevyplňují je. Proto je možnost určení obsahu a významu obrázku obtížnější. Filtrem pro vymezení množiny správných obrázků může být jejich kódování. Pro kódování webové grafiky se nejčastěji využívá rastrové grafiky (`.GIF`) při zachování bezztrátové komprese. Fotografie, loga a jiné obrázky se kódují ztrátovou kompresí a mají převážně koncovky typu .jpg nebo .jpeg. Problémem může být tvar přípony souboru v url adrese. Uvedu příklad: Url adresa má tvar zakončený koncovkou .jpg. Z tohoto tvaru by bylo možné usoudit, že se z webového serveru přenáší komprimovaný obrázek typu jpg. Ve většině případů tomu tak bude, ale existují také případy, kdy se ze serveru vrátí HTML stránka, která svým obsahem poskytuje obrázek. Url adresa může tedy obsahovat celé jméno souboru a přitom se může jednat o webovou stránku. Opačná možnost je v řadě případů také možná a dnes u webových portálů využívaná. Oba problémy je nutné při detekci obrázků uvažovat a v hlavičce HTTP protokolu detekovat reálné kódování.

Jelikož je primárně využíváno bayesovské klasifikace textu, zahrnul jsem při detekci obrázků také kontext obrázku. Kontext se zjišťuje prohledáváním směrem od pozice obrázku až po kořenový uzel dokumentu typu `body`. V souvislostí s klasifikací jsem jako limitní položku stanovil limit 300 znaků ke každému obrázku. Nalezený kontext v tomto rozsahu je vhodný pro další textovou klasifikaci, která se s jeho pomocí upřesňuje. K obrázku a jeho klasifikaci se samozřejmě také přidává titulek a popisek nalezený v atributech.

5.3.4 Aktuálnost

V souvislosti s vyhledáváním relevantních dat jsem také uvažoval nad jejich aktuálností dat. Vzhledem k tomu, že v dnešní době je na internetu většina webových stránek dynamicky generovaných, je obtížné sledovat časové značky, které v případě vygenerování webové stránky na vyžádání neodpovídají skutečnému stáří informací na dané stránce. Změny velikosti stránky také přímo nesouvisí se změnou informace na dané stránce. Z těchto důvodů jsem v řešení neuvažoval sledování časových a velikostních údajů a proto inteligentní klient při zpracování navštěvuje webové stránky opakovaně. Pokud se daný obsah změnil a je odlišný od obsahu v datovém skladu, je uložen. Duplicitní texty nejsou ukládány.

5.3.5 Datový sklad

Datový sklad byl implementován s využitím knihovny jsoncpp. Veškeré informace nalezené a zpracovávané inteligentním klientem jsou ukládány do datových struktur ve formátu typu JSON. V souvislosti s množstvím nalezených informací u skladeb interpretů se datový sklad omezil pouze na uchovávání informací o daném autorovi. Informací o skladbách se na internetu nevyskytují v takové míře jako informace o autorovi. Dodatečné informace ke skladbě nalezneme pouze v případě, že se k ní vztahuje konkrétní historická událost. Proto jsem se v průběhu implementace rozhodl informace o skladbě sloučit s informacemi o autorovi.

Implementace datového skladu byla navržena prostřednictvím struktury zapouzdřených a děděných objektů. Byly vytvořeny bázové objekty pro textové a obrazové informace, které je možné skládat. K jednotlivým objektům je s využitím dědičnosti přidána možnost uchovávání zpětné vazby a zdrojových informací. Každá takto navržená a implementovaná struktura zaručuje v budoucnu možnost vytváření složitějších struktur či uživatelsky definovaných struktur pro ukládání informací.

5.3.6 Klasifikace

Po nalezení souvisejících informací, ať už textového nebo obrazového charakteru se přistupuje ke klasifikaci. Klasifikace vyžaduje uživatelské ohodnocení informací. Pro spuštění klasifikátoru je nutné minimálně jedno označení informace jako správné a jedno označení informace jako nesprávné. Po tomto zásahu si klasifikátor vytvoří lokální databázi znalostí. Z uživatelských zásahů se datbáze nadále rozšiřuje pro možnosti zpřesnění klasifikace. Po každém zásahu tak dochází k doplnění databáze znalostí a reklasifikaci dat.

5.4 Zveřejnění projektu

Pro možnosti dalšího rozšiřování projektu v budoucnosti byl zdrojový kód aplikace zveřejněn na stránkách projektu GitHub[4]. Projekt byl opatřen licencí GNU/GPL v3, která umožňuje svobodnou distribuci a modifikaci programu. GitHub poskytuje git repozitář pro snadné sdílení zdrojových kódů mezi programátory. Repozitář poskytuje zpětnou historii modifikací projektu a umožňuje ostatním provádět změny ve zdrojových kódech. Současně s umístěním zdrojových kódů na GitHub byla vytvořena také webová stránka projektu IMPC[5], kde byl také vytvořen sdílený datový sklad za účelem sdílení informací od klientských aplikací.

[4] https://github.com/tomwagner/IntelligentMPDClient
[5] http://www.impc.cz

Kapitola 6

Implementace serverové části

Z důvodů možností sdílení dat klientů bylo nutné navrhnout a implementovat také serverovou část. Implementace byla provedena v jazyce PHP. Serverová část obsahuje rozhraní, které umožní uživatelům, používajícím inteligentní klienty, vzájemnou výměnu dat. Výměna dat je vhodná zejména pro zpřesnění klasifikace nebo pro sdílení nalezených zdrojů. Zároveň také může ukládat další informace z rozdílných zdrojů, kterých je ve své podstatě neomezené množství.

Serverová část je implementována jako skript v jazyce PHP a složka s právem zápisu webového serveru pro uchování dat ve formátu JSON. Na serverovou část lze také nahlédnout jako na databázi souborů ve formátu JSON, ve které je prováděno vyhledávání a slučování nahrávaných souborů z klientských aplikací. Serverovou část je nutné provozovat na veřejně přístupném webovém serveru s rozhraním do sítě Internet. Pro účely testování a vzájemné distribuce dat jsem tento server umístil a spustil na adrese http://www.impc.cz/api/. Implementace serverové části je navržena tak, aby jí bylo možné využít i jiným inteligentním klientem z různých programovacích jazyků či platforem.

Rozhraní serverové části poskytuje akce:

- `help`,
- `search`,
- `upload`.

První akcí `help` je je triviální výpis nápovědy. Akce `search` přebírá ještě jako parametr `artist` jméno interpreta malými písmeny. Po zaslání parametrů požadavkem GET HTTP protokolu je provedeno lokální vyhledání souboru s autorem a informacemi. Pokud soubor na serveru existuje, je vrácen opět ve formátu JSON zpět k uživateli inteligentního klienta. Na něm je poté interpretován, případně sloučen s lokálním souborem. Pro účely vyhledávání zkomolených jmen interpretů se na serveru vypočítává podobnost s aktuálně uloženými interprety. Klientské aplikaci je poté vrácen záznam, ve kterém byla podobnost nejvyšší. Problém v tomto případě nastává při shodných názvech interpretů, z hlediska řešení a pro účely testování byl tento problém zanedbán.

Poslední akcí serverového rozhraní je akce `upload` pro možnost nahrání souboru z lokální databáze dat uchovávané při běhu klientské aplikace. Pro nahrání souboru se využívá požadavek POST HTTP protokolu. V tomto požadavku lze zaslat libovolně velký textový soubor. Jeho velikost může být omezena pouze stanovením omezení maximální velikosti

nahrávaného obsahu (direktiva `post_max_size`) na server v parametrech konfiguračního souboru `php.ini`. Problémem při řešení serverového rozhraní jsou části lokálního úložiště, které je nutné promítnout do sdíleného prostoru. V případě, že soubor s interpretem na serveru neexistuje, je po nahrání vytvořen a jsou odebrány lokální parametry, které nejsou pro sdílení podstatné (např. zda uživatel daný objekt již hodnotil, zda byl objekt se sdíleným úložištěm synchronizován apod.). Jestliže po nahrání uživatelova lokálního souboru již soubor s tímto jménem ve sdíleném úložišti existuje, je nutné provést sjednocení. Při sjednocování dvou souborů je nutné provádět slučování pro každý objekt samostatně, jelikož je zohledňováno uživatelské hodnocení, které je potřeba ve sdíleném úložišti uchovávat. Dle implementace by každý uživatel měl mít možnost objekt hodnotit a s tímto hodnocením se „podělit“ v rámci sdíleného úložiště. Každý uživatel má možnost hodnocení. Zda jej provedl, a záznam byl již synchronizován, určuje hodnota booleovské proměnné udržované u každého objektu v lokálním datovém úložišti. Server také zpracovává případné duplicity objektů, které jsou řešeny za pomocí hashování.

Kapitola 7

Testování

7.1 Klasifikace

Testování klasifikace je komplikovaná a časově náročná činnost. Pro zhodnocení klasifikátoru jsem vymyslel dva testy. První test viz. tabulka č.7.1 zobrazuje měření po stažení dat a ohodnocení 5 záznamů jako správných a 5 záznamů jako nesprávných. Jelikož při hledání dat je zohledněna přítomnost jména interpreta v daném dokumentu, je možné, jako tomu bylo u vzorku Elvise Presleyho, že ani 5 špatných vzorků nebylo v datech přítomno. Z výsledků klasifikace u tohoto záznamu je patrné, že klasifikace vycházela z nedostatku ohodnocení a proto jsou procenta úspěšnosti nízká. Naopak u vzorků kapely Europe, kde se do výsledků zařadily i výsledky pro evropský kontinent je úspěšnost klasifikace a rozpoznání velmi vysoká. Sloupce T/F značí poměr správně a nesprávně (True/False) klasifikovaných záznamů. Poslední dva sloupce zobrazují procentuální úspěšnost klasifikace při ohodnocených záznamech.

Interpret	**text**	**obrázky**	**text T/F**	**obr. T/F**	**úsp. text**	**úsp. obr.**
Abba	39 ks	76 ks	35/4	70/6	89,74%	92,10%
Beatles	34 ks	76 ks	29/2	65/11	85,29%	85,53%
Europe	32 ks	53 ks	29/3	49/4	90,62%	92,45%
Elvis Presley	21 ks	51 ks	16/5	44/7	76,19%	86,27%

Tabulka 7.1: Tabulka měření úspěšnosti klasifikace po ohodnocení záznamů.

V testování byly provedeny i testy, které zahrnují syntakticky stejného interpreta avšak se sémanticky odlišným významem. Jednalo se o kapelu Europe. Ze zdrojů jako Wikipedie a Google byly podávány informace jak o kapele tak o kontinentu Evropě a její historii. V těchto testech se bayesovská klasifikace osvědčila, po ohodnocení prvního článku o Evropě bylo 77% záznamů souvisejících s Evropou překlasifikováno jako nerelevantních a byl rozpoznán správný význam v souvislosti s hudební skupinou, na kterou byl dotaz položen.

7.2 Učení a zdokonalování

Jako druhý test jsem se pokusil zpracovat test ve kterém by měla být patrná schopnost aplikace učit se a zdokonalovat se. Test spočíval ve stažení informací o interpretovi ze tří hudebních databází. Konkrétně databází last.fm, allmusic.com a discogs.com. Po stažení

dat byla data ručně hodnocena. V dalším kroku, kdy byla již známa databáze znalostí z klasifikovaných dat, byl tento pokus proveden při vyhledání autora v google.com. V posledním kroku bylo zopakováno to samé ale ve vyhledávači yahoo.com. Test byl zaměřen hlavně na anglicky psané texty o autorech.

K následujícím tabulkám připojuji vysvětlivky:

- **T** - Text - celkový počet textových informací - neklasifikovaných,
- **I** - Images - celkový počet obrazových informací - neklasifikovaných,
- **TT** - Text True - počet nalezených správně klasifikovaných textových informací,
- **TF** - Text False - počet nalezených chybně klasifikovaných textových informací,
- **IT** - Image True - počet nalezených správně klasifikovaných obrazových informací,
- **IF** - Image False - počet nalezených chybně klasifikovaných obrazových informací,
- **ST** - Success Text - procentuální úspěšnost klasifikace textových informací,
- **SI** - Success Images - procentuální úspěšnost klasifikace obrazových informací,
- **FI** - False Images - počet klasifikovaných obrazových informací jako nesprávných,
- **FT** - False Text - počet klasifikovaných textových informací jako nesprávných.

Interpret	**T**	**I**	**TT**	**TF**	**IT**	**IF**
Adele	13	48	0	0	0	0
Al Jarreau	15	111	0	0	0	0
Kiss	18	138	0	0	0	0
The Beatles	16	174	0	0	0	0

Tabulka 7.2: 1. krok - učení a vytvoření databáze znalostí

Interpret	**TT**	**TF**	**IT**	**IF**	**FT**	**FI**	**ST**	**SI**
Adele	28	1	33	3	29	41	96,55%	91,66%
Al Jarreau	44	4	32	3	4	0	91,66%	91,43%
Kiss	15	5	7	1	42	46	75%	87,50%
The Beatles	25	6	39	4	35	6	80,64%	90,70%

Tabulka 7.3: 2. krok - test klasifikace s naučenou databází znalostí na výsledcích z google.com

Při testech jsem cíleně vybral dva interprety se jménem, které má více významů (skupina Kiss a zpěvačka Adele). U nich lze pozorovat, že nalezené informace měly vysokou hodnotu nesprávně klasifikovaných textových i obrazových informací (sloupce FI a FT tab. č. 7.3 a 7.4). Naopak u druhé dvojice interpretů jsou hodnoty výrazně nižší. Z testů klasifikace bohužel nelze vyvodit konečný a jednoznačný závěr. Výsledné hodnoty klasifikace mohou být zkresleny nižším počtem zkoumaných informací. Pro přesnější a sofistikovanější testování

Interpret	TT	TF	IT	IF	FT	FI	ST	SI
Adele	25	4	31	5	8	68	86,21%	86,11%
Al Jarreau	6	1	46	9	1	2	85,71%	83,67%
Kiss	3	0	4	1	3	96	100%	80%
The Beatles	9	2	27	4	15	20	81,81%	87,10%

Tabulka 7.4: 3. krok - test klasifikace s naučenou databází znalostí na výsledcích z yahoo.com

by bylo nezbytné využít větší naučené databáze znalostí a manuálně hodnotit množství dat v řádech stovek položek. Toto hodnocení není proveditelné v reálném čase jedním člověkem. Strojové zpracování bohužel také nepřichází v úvahu, jelikož neexistuje algoritmus, který by posoudil úroveň souvislosti informace. K hodnocení velkého množství dat je tak nutné využít jedině činnosti hodnotitelů, což se v případě této práce může projevit až po delším používání aplikace s množstvím hodnotitelů, kteří budou ochotni se o ohodnocené informace podělit.

7.3 Výkonnostní test

V souvislosti s obecnými testy na funkčnost jsem provedl také jeden test na výkon. Při hledání informací o kapele The Beatles. Za dobu přehrání celé skladby (175 sekund) bylo zaznamenáno 608 stažených webových stránek, 1758 článků (dvojice nadpis a odstavec), z nichž 608 odpovídalo rozsahu odstavce v minimální délce 300 znaků. Bylo nalezeno 1902 obrázkových souborů z nichž 856 splňovalo požadavky na obrázek v minimální délce jedné ze stran 100px. Lokální cache v tomto případě obsahovala 1449 položek souborů a obrazových materiálů o celkové velikosti 25.7 MB.

Kapitola 8

Závěr

V práci jsem se zabýval teoretickým rozborem, návrhem a implementací inteligentní klientské aplikace pro hudební přehrávací server MPD. Inteligence klientské aplikace spočívá v prezentaci souvisejících a relevantních informací k přehrávané skladbě. Inteligence je navíc rozšířena o možnosti učení, kdy se inteligentní klient prostřednictvím uživatelských zásahů učí lépe rozpoznávat a hodnotit relevantní obsah.

V teoretickém rozboru jsem popsal problematiku agentních systémů a učícího se agenta, který je jedním z požadavků kladených na inteligenci klientské aplikace. K tomu aby se agent orientoval ve webovém prostředí je nezbytná klasifikace dat. Proto bylo v teoretické části rozebráno více metod klasifikace, z nichž se jako nejvhodnější pro návrh a implementaci ukazuje Bayesovská klasifikace, často využívaná například pro filtrování nevyžádaných emailových zpráv. Nedílnou součástí je také souhrn značkovacích jazyků, používaných v současném webovém prostředí internetu, a popis komunikačního protokolu MPD serveru.

Práce se věnovala také rozboru stávajících klientských aplikací. Bylo vybráno několik aplikací, které splňují požadavky na prezentaci souvisejících metadat. Z rozboru aplikací vyplynulo, že v sobě neukrývají v podstatě žádnou úroveň inteligence. Přistupují do ověřených hudebních databází a prostřednictvím aplikačního rozhraní vybírají vhodná data k přehrávanému obsahu. Úroveň správnosti prezentovaných dat je tak vyšší, než při shromažďování informací z více neověřených zdrojů. Proto také nezahrnují uživatelské hodnocení obsahu, které v případě zobrazování ověřených dat není žádoucí.

Po teoretickém rozboru byl proveden návrh prostřednictvím schématu vzájemně komunikujících modulů. Ty představují nejdůležitější části klientské aplikace. Bylo definované multiagentní prostředí, návrh GUI a návrh struktury databáze znalostí. V návrhu je také popsáno, jak efektivně vyhledávat informace v dokumentech popsaných značkovacím jazykem HTML. Inspirace pochází především z práce robotů, kteří indexují obsah pro vyhledávače. Současně zde vznikla myšlenka na urychlení vyhledávání informace v příslušném webovém zdroji. Agent by měl mít schopnost automaticky nalézt formulář pro vyhledávání na webové stránce. Zadáním vyhledávaného dotazu dojít až k požadovanému obsahu. Nebylo by tedy nutné přistupovat k rekurzivnímu sestupu prostřednictvím hypertextových odkazů.

Implementace jsem rozdělil na klientskou a serverovou část. Obě části byly implementovány tak, jak bylo popsáno v návrhu. Implementace klientské části zahrnula bayesovský klasifikátor jako prostředek k dosažení učících schopností a požadované míry inteligence. Navrhovaná schopnost vyhledání vyhledávacího formuláře jednoznačně urychlila průchod jednotlivých zdrojů. V přístupu do webových hudebních databází byla velká část průchodů učiněna právě prostřednictvím vyhledávacího pole. Stejně tak, jako by tomu bylo při hledání

relevantních informací člověkem. Při hledání textových informací na webových stránkách se však při identifikaci užitečného textu vyskytly první problémy, se kterými se v návrhu nepočítalo. Tyto problémy byly vyřešeny přiřazováním textu k nejbližším nadpisům. Metoda detekce textu na stránce nemusí být v případě nestrukturovaného textu webové stránky plně funkční. Metody pro detekci obrázků jsou v implementaci funkční. Pro klasifikaci obrázků prostřednictvím textového klasifikátoru byl využit okolní kontext obrázku.

V závěrečném testování byly provedeny tři testy. První test zahrnuje testování klasifikátoru, kde se úspěšnost klasifikace pohybovala v rozmezí 85-92%. Druhý test se zaměřoval na učící a zdokonalující se schopnosti bayesovské klasifikace. Učící schopnosti se ale bohužel prokázat nepodařilo. Chyba dle mého názoru vznikla nízkým počtem naučených dat obsažených v databázi znalostí. Jelikož je množství naučených znalostí závislé na člověkem udávaném hodnocení, není v časových možnostech jedné osoby provést tato měření, která by měla čítat stovky až tisíce vzorků dat. Poslední test z řady testů ukazuje výkonnost klientské aplikace při hledání dat ve webovém prostředí.

8.1 Přínos práce

Podařilo se mi implementovat unikátní klientskou aplikaci pro hudební přehrávací server MPD. Unikátní v možnostech, které poskytuje a nabízí uživateli. Doposud žádná aplikace se v této oblasti nevytvářela. Proto byl jak návrh tak i konečná implementace velmi náročná. Implementace byla navržena pro možnosti dalšího rozšiřování, případně převzetí části zdrojových kódů. Z těchto důvodů byla aplikace licencována licencí GNU GPL v3, která umožňuje bezplatné a svobodné nakládání se zdrojovými kódy. Zároveň byla aplikace zveřejněna na distribučním serveru GitHub. Možností jak aplikaci rozšířit je celá řada, od volby užšího zaměření na konkrétní výběr dat, přes možnost implementace jiného druhu klasifikátoru dat, až po vylepšení grafické podoby úrovně prezentace.

Literatura

[1] Index of the HTML 4 Elements. 1999, [Online; navštíveno 20.10.2011].
URL http://www.w3.org/TR/html4/index/elements.html

[2] XHTML 1.0: The Extensible HyperText Markup Language (Second Edition). 2002, [Online; navštíveno 21.10.2011].
URL http://www.w3.org/TR/xhtml1/#general

[3] W3C: Document Object Model (DOM). 2004, [Online; navštíveno 21.10.2011].
URL http://www.w3.org/TR/DOM-Level-3-Core/

[4] ncmpc++ mpd client. 2010, [Online; navštíveno 26.10.2011].
URL http://unkart.ovh.org/ncmpcpp/

[5] Chakrabarti, S.: *Mining the Web - Discovering Knowledge from Hypertext Data*. San Francisco: Morgan Kaufmann Publishers, 2003, ISBN 978-1-55860-754-5, 345 s.

[6] Community, M.: Protocol Reference - Music Player Daemon Community Wiki. 2011, [Online; navštíveno 28.11.2011].
URL http://mpd.wikia.com/wiki/Protocol_Reference

[7] Crockford, D.: rfc4627:The application/json Media Type for JavaScript Object Notation (JSON). 2006, [Online; navštíveno 10.11.2011].
URL http://www.ietf.org/rfc/rfc4627.txt?number=4627

[8] Hordějčuk, V.: Neuronové sítě. 2011, [Online; navštíveno 1.1.2012].
URL http://old.voho.cz/wiki/neuronove-site/

[9] Jakel, M.: Stavové kódy a hlášení v odpovědi protokolu HTTP. 2002, [Online; navštíveno 17.10.2011].
URL http://interval.cz/clanky/stavove-kody-a-hlaseni-v-odpovedi-protokolu-http/

[10] Jiawei Han, J. P., Micheline Kamber: *Data Mining: Concepts and Techniques, Second Edition*. San Francisco: Morgan Kaufmann Publishers, 2006, ISBN 978-1558609013, 743 s.

[11] R. Fielding, J. M., J. Gettys: Hypertext Transfer Protocol – HTTP/1.1. 1999, [Online; navštíveno 5.11.2011].
URL http://tools.ietf.org/html/rfc2616

[12] Rescorla, E.: Protocol Reference - Music Player Daemon Community Wiki. 2000, [Online; navštíveno 5.11.2011].
URL http://tools.ietf.org/html/rfc2818

[13] Steven Holzner, J. Z.: *Mistrovství v AJAXu.* Brno: Computer Press, 2007, ISBN 978-80-251-1850-4, 591 s.

[14] Stuart J. Russell, P. N.: *Artificial Intelligence: A Modern Approach.* New Jersey: Prentice Hall, 1995, ISBN 0-13-103805-2, 960 s.

Příloha A

Obsah CD

- zdrojové kódy inteligentního klientského přehrávače
- zdrojové kódy serverového rozhraní + popis serverového rozhraní
- pdf dokumentace obsahující návod k instalaci
- zdrojový text diplomové práce ve formátu LaTeX
- text diplomové práce ve formátu PDF