Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Webová služba pro doporučování hudby

2018

Bc. Jiří Valůšek

Vedoucí práce:
Mgr. Martin Trnečka, Ph.D.

Studijní obor: Informatika, prezenční forma

Bibliografické údaje

Autor:	Bc. Jiří Valůšek
Název práce:	Webová služba pro doporučování hudby
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2018
Studijní obor:	Informatika, prezenční forma
Vedoucí práce:	Mgr. Martin Trnečka, Ph.D.
Počet stran:	44
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Bc. Jiří Valůšek
Title:	Web services for music recommendation
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2018
Study field:	Computer Science, full-time form
Supervisor:	Mgr. Martin Trnečka, Ph.D.
Page count:	44
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

V této práci si přiblížíme problematiku doporučování hudby. Navrhneme algoritmus, který implementujeme formou webové aplikace. Hudba bude doporučována pomocí přehrávání nekonečného seznamu skladeb. Algoritmus bude dbát na individuálnost doporučení, typ a náladu posluchače a zamezení častého opakování skladeb.

Synopsis

In this thesis we will focus on music recommendation. We will create an algorithm which will be implemented as a web application. The music will be recommended as stream of tracks. The algorithm will recommend individual content based on type and mood of a listener. Also, it will restrict frequent played tracks.

Klíčová slova: doporučování, hudba, algoritmus, webová aplikace

Keywords: recommendation, music, algorithm, web application

Děkuji vedoucímu práce doktoru Martinu Trnečkovi za rady a vedení práce.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam vět

1 Úvod

Hudební průmysl se za poslední léta razantně změnil. Časy, kdy se ve velkém prodávaly CD nosiče, jsou pryč. S rozšířením internetu se hudba začala prodávat prostřednictvím online obchodů. Ovšem se zvyšováním rychlosti připojení se objevil nový typ služby, který naprosto změnil způsob konzumace hudebního obsahu. Jsou to streamovací služby. Ty uživateli za měsíční poplatek zpřístupní katalog s nepřeberným množstvím hudby.

Další velkou změnou, kterou tyto služby přinesly, je individuálnost obsahu. Lidé nemusí poslouchat rádio, tak jak ho známe z minulosti (tzn. lineárně vysílaný stream, který je pro všechny stejný). Nyní si každý uživatel může pustit vlastní rádio, které mu individuálně pouští doporučenou hudbu. A právě doporučování hudby se budeme na následujících stránkách věnovat.

Nejprve se podíváme na problematiku doporučovacích systémů obecně a následně se zaměříme speciálně na doporučování hudby, které nese svá specifika. Neopomeneme ani přehled současných hudebních online služeb.

Našim hlavním cílem je vytvořit vlastní algoritmus pro doporučování hudby. Tento algoritmus bude realizován prostřednictvím webové aplikace. Bude se jednat o takové rádio, jak ho známe ze streamovacích služeb. To znamená, že uživateli bude předkládán nekonečný seznam doporučených skladeb.

Náš algoritmus by měl splňovat několik základních podmínek:

- Individuálnost doporučení: Ke každému uživateli by se mělo přistupovat individuálně. Na základě jeho hodnocení skladeb, by se mělo doporučení měnit.
- Zohlednění nálady: Každý nemá neustále stejnou náladu. Algoritmus by měl dát uživateli možnost nastavit nějaké vstupní parametry.
- Zohlednění typu posluchače: Zájem o hudbu se u každého liší. Algoritmus by na to měl být připraven.
- Zamezení častému opakování: Někdy není příjemné, když se doporučení často opakují. Mělo by být zajištěno, aby k tomu nedocházelo.

Vzhledem k tomu, že uživatelé budou doporučované skladby hodnotit, by měla webová aplikace umožňovat tato hodnocení kdykoliv zpětně měnit. Rovněž by neměla chybět možnost vyhledání interpreta, alba nebo skladby.

2 Doporučovací systémy

Intuitivně můžeme problém doporučení rozdělit na dva podproblémy (viz [1]). Prvním je problém predikce, který představuje pravděpodobnost, že se daná položka bude uživateli líbit. Druhým podproblémem je doporučení seznamu N položek, které by se mohli uživateli nejvíce líbit.

Tyto podproblémy můžeme formalizovat takto (viz [1]):

Definice 1 (Problém predikce)
Nechť $U = \{u_1, u_2, ...u_m\}$ je množina všech uživatelů a $I = \{i_1, i_2, ...i_n\}$ je množina všech položek, které mohou být doporučeny.

Každý uživatel u_i má seznam položek I_{u_i}. Tento seznam reprezentuje položky, k nimž uživatel projevil zájem. Poznamenejme, že množina $I_{u_i} \subseteq I$ a může nastat $I_{u_i} = \emptyset$.

Potom funkce $P(u_a, i_j)$ je pravděpodobnost, že se položka i_j bude líbit aktivnímu uživateli u_a tak, že $i_j \notin I_{u_a}$.

Definice 2 (Problém doporučení)
Problém doporučení je množina I_r, obsahující N položek s nejvyšší hodnotou $P(u_a, i_j)$ taková, že $I_r \subset I$ a $I_r \cap I_{u_i} = \emptyset$.

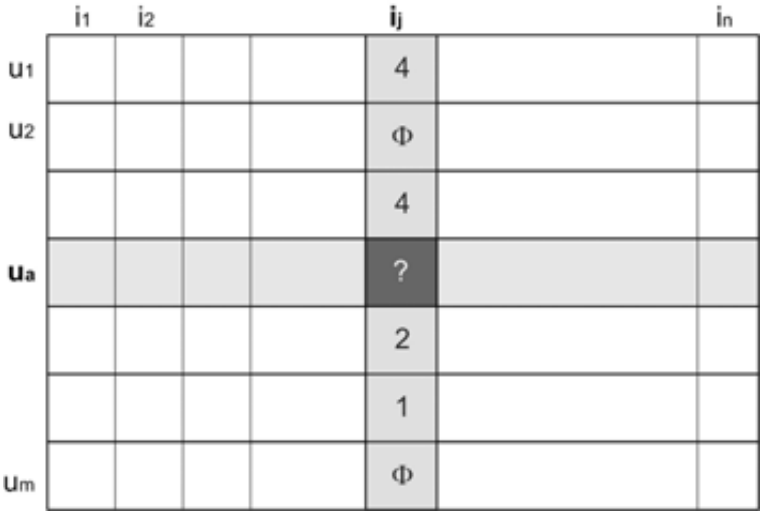
Ve většině doporučovacích systémů je pravděpodobnostní funkce reprezentována hodnocením. To představuje trojice $\langle u, i, r \rangle$, kde uživatel u hodnotí položku i hodnotou r. Běžně bývá hodnota r reálné číslo v intervalu mezi 0 a 1, číslo od 1 do 5 nebo líbí/nelíbí.

2.1 Metody doporučování

Pokud chceme vytvořit systém pro doporučování, máme k dispozici několik možností (viz [1]). Můžeme vycházet z historie hodnocení položek uživatelem, z informací v jeho profilu, z dat u položky nebo dokonce tyto možnosti zkombinovat.

2.1.1 Demographic Filtering

Demographic Filtering (DF) je metoda, která doporučuje položky podle klasifikace uživatele. Na základě dat, jako jsou osobní informace (věk, rodinný stav, pohlaví a další), polohová data (město, stát) a psychografická data (zájmy, životní styl a další), je uživatel zařazen k určitému stereotypu. Následně jsou doporučovány položky s ohledem na daný stereotyp. Hlavní nevýhodou této metody je, že doporučuje položky lidem s podobnými demografickými informacemi, což je příliš obecné. Celkem vzato se jedná o nejjednodušší metodu.



Obrázek 1: Matice uživatelů a položek (převzato z [1]).

2.1.2 Collaborative Filtering

Collaborative Filtering (CF) predikuje preference uživatele k položkám z jeho interakce v minulosti. To znamená, že uživatel dává zpětnou vazbu systému prostřednictvím hodnocení položek a ten díky těmto informacím navrhuje položky. K vytvoření doporučení tato metoda využívá matice M s m řádky reprezentujícími uživatele a n sloupci reprezentujícími položky. Hodnota M_{u_a,i_j} je hodnocení položky i_j uživatelem u_a. Matice je znázorněná na obrázku 1.

Jak ale tuto matici zpracovat tak, abychom dostali doporučení? Existuje několik různých technik, které můžeme rozdělit do tří kategorií [2]:

1. Memory-Based,
2. Model-Based,
3. Hybrid.

Do první kategorie spadají techniky založené na výpočtu podobnosti dvou položek, nebo uživatelů (Item-Based a User-Based). Jednou z jejich hlavních výhod je jednoduchá implementace. U doporučovacích systémů často dochází k problému s řídkostí matice. V těchto případech je lepší použít Model-Based techniky, mezi které patří např. Shape-One CF, redukce dimenzí pomocí faktorizace matice (Singular Value Decomposition, Non-negative Matrix Factorization nebo Principal Conponent Analysis). Avšak jednou z jejich nevýhod je složitost vytvoření modelu. Poslední kategorie kombinuje techniky z obou předchozích, díky tomu řeší některé obecné problémy CF jako řídkost nebo *gray sheep* problém (viz níže).

CF je sice jednou z nejpoužívanějších, má ovšem několik nevýhod:

- Řídkost dat a velká dimenze: Při relativně velkém počtu uživatelů a položek je problémem malé pokrytí položek uživatelským hodnocením. To může způsobovat obtížné nalezení kvalitního sousedního uživatele.
- Atypický vkus: Uživatelé, kteří mají velmi odlišný vkus, budou mít málo sousedních uživatelů. Důsledkem bude špatné doporučení. Tento problém je také znám jako *gray sheep* problém.
- Cold-start: Když je do systému přidán nový uživatel/položka, je pro systém těžké jej zařadit, protože má málo hodnocení.
- Pouze zpětná vazba: Metoda CF je postavena pouze na zpětné vazbě uživatelů (hodnocení, stažení, objednání) a nebere v úvahu vlastnosti položek. Je to subjektivní metoda, která vytváří sociální sousedy, což běžně způsobuje doporučování nejpopulárnějších položek.
- Populární se stává populárnější: Populární položky mají více hodnocení, čímž nastává situace, že jsou i častěji doporučovány. Někdy je ovšem pro uživatele lepší méně populární a zajímavější položka nebo také položka nová.
- Feedback loop: Uživatelé se prostřednictvím hodnocení navzájem ovlivňují. To znamená, že ti, kteří jsou v systému dříve ovlivní doporučení těm, kteří do něj vstoupí později.

2.1.3 Content-Based Filtering

Content-Based Filtering (CB) k doporučování využívá charakteristiky položek. Z preferencí uživatele následně určí položky, které jsou mu nejblíže. Charakteristiky mohou být generovány automaticky (např. analýzou) nebo manuálně odborníkem. Klíčovou součástí této metody je podobnostní funkce. Jedná se o funkci, která počítá vzdálenost mezi dvěma položkami. Většina metrických funkcí pracuje s číselnými hodnotami (vektory). Nyní si některé z nich uvedeme:

Definice 3 (Euklidovská metrika)

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Definice 4 (Manhattanská metrika)

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Definice 5 (Chebychevova metrika)

$$d(x, y) = max_{i=1..n}|x_i - y_i|$$

Definice 6 (Mahalanobisova metrika)

$$d(x, y) = \sqrt{(x - y)^T S^{-1}(x - y)}$$

Jako předchozí metody i tato má své nevýhody:

- Cold-start: Na rozdíl od CF se tento problém týká pouze uživatelů.
- Atypický vkus: I u této metody může docházet k problému *gray sheep*.
- Problém novosti: Za předpokladu, že podobnostní funkce pracuje správně, systém uživateli vždy doporučí položky podobné těm, co už se mu líbily. To brání doporučení položek s jinou charakteristikou, které by se uživateli mohly také líbit.
- Složitost domény: Někdy je těžké vyčíslit vlastnost tak, aby s ní podobnostní funkce uměla pracovat. Může jít např. o analýzu nálady písničky.
- Podobnost pouze na základě vlastností: Podobnostní funkce pracuje pouze s vlastnostmi, ale nebere v úvahu subjektivnost nebo osobní názory.

2.1.4 Context-Based Filtering

Kontext je jakákoliv informace, která popisuje situaci nebo entitu. Metoda Context-Based Filtering (CT) využívá kontextové informace k popisu a charakterizování položek. K získání kontextuálních informací se využívá např. sociálního značkování, kde uživatelé označí položku nějakým klíčovým slovem. To ovšem může způsobit několik problémů. Nejvíc značek budou mít nejpopulárnější položky. Pokud není seznam značek uzavřená množina, může docházet k problému s podobnými slovy. Jako poslední příklad nevýhod uvedeme efekt, kdy uživatelé k položkám přidávají cíleně špatné značky.

2.1.5 Hybridní metoda

Hlavní myšlenkou této metody je vytvoření lepšího doporučovacího systémů kombinací některých předchozích metod. Díky tomu můžeme vyřešit nevýhody jednotlivých metod. Podrobnější informace je možné najít např. v [3].

3 Doporučování hudby

V následující kapitole se budeme blíže soustředit na oblast hudby. Oproti obecnému systému doporučování má svá specifika. Začneme případy užití doporučovacího systému a budeme pokračovat přehledem několika služeb, které tento systém v současnosti využívají.

3.1 Případy užití

Hlavním cílem hudebního doporučovacího systému je navrhnout zajímavou hudbu vzhledem k uživatelskému profilu. Měla by se skládat ze známých i neznámých skladeb, alb nebo umělců. Většina systémů se zaměřuje na navrhování seznamu umělců nebo neuspořádaného proudu skladeb. Jsou zde ale i jiné zajímavé možnosti, jak hudbu doporučovat (viz [1]).

3.1.1 Doporučení umělců

Doporučení umělců je možné na základě uživatelského profilu. Na rozdíl od jiných typů položek si však nese i další výhody. Je zde možné využít i informací jako např. nově vydaná alba, seznam koncertů, související zprávy, recenze a další.

3.1.2 Generování playlistů

Hudbu můžeme doporučovat i prostřednictvím playlistů. Nezáleží u nich na pořadí skladeb (jako u mixu), ale zaměřují se spíše na určitý emocionální stav nebo situaci (např. v práci, při běhání nebo relaxaci). Máme několik možností, jak tyto playlisty vygenerovat:

- čistě náhodně,
- na základě skladby nebo interpreta,
- z informací v profilu.

3.1.3 Rádio

Streamovací služby přinesly myšlenku rádia. Tento typ doporučení je podobný generování playlistů, jen se není možné v přehrávání vracet a seznam písniček je nekonečný.

3.1.4 Doporučování prostřednictvím sousedů

Objevovat novou hudbu je také možné prostřednictvím nejbližších sousedů. To znamená těch, kteří jsou uživateli vkusem nejblíže. Není to ale pravidlem. V posledních letech se stále více objevuje doporučování hudby, kterou poslouchají přátelé.

3.2 Služby pro doporučování hudby

V současnosti existuje několik velmi populárních služeb, které systém doporučování hudby využívají. My si uvedeme jejich čtyři zástupce.

3.2.1 Spotify

Největší hudební streamovací službou je aktuálně švédské Spotify [4]. Používá ho 159 milionů uživatelů. Spotify je možné poslouchat bezplatně (s reklamou), nebo v placeném režimu. K dispozici je několik různých balíčků: pro jednotlivce, pro rodiny a pro studenty. Momentálně má Spotify 71 milionů platících uživatelů [5].

K dispozici je obsáhlý katalog hudby, který je možné jednoduše prohledávat. V každém profilu umělce je možné najít i doporučení dalších, jež s ním souvisí. Nechybí ani informace o nadcházejících koncertech. Spotify myslí i na vzájemné vazby přátel, kteří si mezi sebou mohou sdílet hudbu.

Kromě ručního procházení databáze a naplňování si své hudební knihovny, Spotify umožňuje poslouchat hudbu formou rádia. To může hrát buď podle zvoleného žánru nebo umělce. Hudbu ale nedoporučuje pouze formou rádií. Služba obsahuje několik playlistů roztříděných do žánrů a nálad. Vše je doplněno playlisty, jako *Discover Weekly*, které se periodicky generují v čase přímo pro konkrétního uživatele. Právě *Discover Weekly* používá algoritmus *Release Radar* [6], který kromě doporučovacího systému využívá i umělé inteligence.

K doporučování používá Spotify CF, řešený implicitní faktorizací matice. Navíc využívá metadata, analýzu textu i audia [6].

3.2.2 Apple Music

Druhou největší službou je s 38 miliony předplatiteli [7] Apple Music [8]. Na rozdíl od Spotify ale není bezplatný. Stejně jako jeho konkurence nabízí přístup k rozsáhlému katalogu hudby. Další interprety také doporučuje prostřednictvím sekce *Podomní umělci* v jejich profilech. Nepříjemností ale je, že se v katalogu objevují profily, které obsahují hudbu různých interpretů se stejným jménem. Pokud by tyto informace využíval algoritmus doporučení hudby, nepracoval by úplně korektně. Představme si situaci, kdy by dva umělci s totožným jménem produkovali razantně odlišný styl hudby. Veškerá jejich tvorba by ale byla v jednom profilu a pro algoritmus by byli jako jeden. Výsledná doporučení by nebyla tolik přesná, protože by nevycházela z reálného stavu. To by samozřejmě neplatilo, pokud by se jednalo o stejného umělce, který v průběhu své kariéry zásadně změnil hudební styl.

Apple Music také doporučuje hudbu prostřednictvím rádia. Na rozdíl od Spotify ale navíc obsahuje i moderované rádio *Beats 1*, které vysílá pro všechny posluchače stejně a je tedy jako rádia, která známe z klasického FM vysílání. Apple nezapomíná ani na doporučování hudby formou generování playlistů. Příkladem je personalizovaný playlist *Favourites Mix*, který se opakovaně generuje každý týden.

3.2.3 Youtube

Youtube [9] je sice služba na sdílení videí, přesto se zaměřuje i na hudbu a to hlavně prostřednictvím kanálu *VEVO*. Novou hudbu je možné objevovat díky mixovaným playlistům, které jsou generovány podle konkrétní písničky nebo interpreta. Pokud se přehrává video, je možné pomocí tlačítka další (nebo počkat po konci přehrávání) přeskočit na další v pořadí. To se bohužel nedá úplně srovnat s rádii od Spotify nebo Apple Music, protože se často stává, že se videa začnou cyklicky opakovat.

Youtube pro personalizovaná doporučování využívá systém zvaný *RS* [6]. Ten je poháněn umělou inteligencí Google Brain [10] a skládá se ze dvou neuronových sítí. První sleduje historii zhlédnutí a používá CF k výběru stovek videí. Tento proces známý jako candidate generation využívá zpětnou vazbu uživatelů k trénování modelu. Druhá neuronová síť řadí videa tak, aby uživateli poskytla doporučení.

3.2.4 Last.fm

Last.fm [11] je sociální síť pro doporučování hudby. Lidé si na ni mohou zaznamenávat hudbu, kterou poslouchají na jiných službách (Spotify, Apple Music atd.). Služba mimo jiné umožňuje nalezení uživatelů s podobným hudebním vkusem.

Datovou síť Last.fm si můžeme představit jako graf se dvěma druhy hran [12]. První spojuje uživatele s uživatelem a druhý uživatele s umělcem. První druh hrany může spojovat uživatele, kteří jsou na síti přáteli, nebo uživatele, kteří jsou si podobní z hlediska hudebního stylu. Druhý druh hrany může mít silnější/slabší spojení, v závislosti na frekvenci poslouchání daného umělce. K doporučování last.fm využívá CF obohacený o práci se sociálním propojením uživatelů. To znamená, že nedoporučuje jen ty umělce, které poslouchá uživatel s blízkým hudebním vkusem, ale také umělce, které poslouchají jeho přátelé.

4 Prerekvizity

Než se pustíme do algoritmu, je potřeba si projít dvě teoretické oblasti, které při tvorbě algoritmu využijeme. První je pravděpodobnost a druhou The Long Tail problém.

4.1 Pravděpodobnost

Jak formálně definovat pojem pravděpodobnost, který známe z běžného života? Označme množinu všech možných výsledků nějakého náhodného pokusu Ω. Prvky této množiny se nazývají elementární jevy a budeme je značit ω. Jev je libovolná podmnožina Ω. Než přistoupíme k definici pravděpodobnosti, definujeme pojem σ-algebra.

Definice 7 (σ-algebra)
σ-algebra na neprázdné množině Ω je neprázdná podmnožina $\mathcal{A} \subseteq 2^{\Omega}$ splňující následující podmínky:

1. je-li $A \in \mathcal{A}$, pak $\neg A \in \mathcal{A}$;
2. jsou-li $A_1, A_2, ... \in \mathcal{A}$, pak $\cup_{i=1}^{\infty} A_i \in \mathcal{A}$.

Nyní známe vše potřebné k tomu, abychom mohli definovat pravděpodobnostní prostor a pravděpodobnostní funkci P.

Definice 8 (Pravděpodobnostní prostor)
Nechť $\langle \Omega, \mathcal{A} \rangle$ je σ-algebra na Ω. Pravděpodobnostní prostor je uspořádaná trojice $\langle \Omega, \mathcal{A}, P \rangle$, kde P je funkce splňující:

1. $P(A) \geq 0$ pro každý $A \in \mathcal{A}$,
2. $P(\Omega) = 1$,
3. $P(\cup_{i=1}^{\infty} A_i) = \Sigma_{i=1}^{\infty} P(A_i)$ pro každou posloupnost jevů $A_1, A_2, ...$, které jsou po dvou disjunktní, tj. $A_i \cap A_j = \emptyset$ pro $i \neq j$.

K lepší práci s pravděpodobností se využívá pojem náhodné veličiny.

Definice 9 (Náhodná veličina)
Nechť $\mathcal{A}$ je σ-algebra jevů na množině Ω elementárních jevů. $\mathcal{B}$ nechť je borelovská σ-algebra na $\mathbb{R}$. Funkce $X : \Omega \to \mathbb{R}$ se nazývá náhodná veličina, pokud pro každou množinu $B \in \mathcal{B}$ platí $X^{-1}(B) \in \mathcal{A}$, kde

$$X^{-1}(B) = \{\omega \in \Omega | X(\omega) \in B\}.$$

Náhodná veličina je tedy funkce, která mapuje prvky množiny elementárních jevů do množiny reálných čísel.

Definice 10 (Distribuční funkce)
Nechť $\langle \Omega, \mathcal{A}, P \rangle$ je pravděpodobnostní prostor a $X : \Omega \to \mathbb{R}$ náhodná veličina. Funkce $F : \mathbb{R} \to [0, 1]$ definovaná

$$F(x) = P(\{\omega \in \Omega | X(\omega) \leq x\})$$

se nazývá distribuční funkce náhodné veličiny X. Někdy se bere $F(x) = P(\{\omega \in \Omega | X(\omega) < x\})$.

Existují dva druhy náhodných veličin: diskrétní a spojité.

Definice 11 (Diskrétní náhodná veličina)
Náhodná veličina se nazývá diskrétní, pokud nabývá jen konečně nebo spočetně mnoha hodnot.

Definice 12 (Pravděpodobnostní funkce diskrétní náhodné veličiny)
Funkce přiřazující hodnotám $x \in \mathbb{R}$ pravděpodobnosti $P(A_x)$ se značí p_X a nazývá se pravděpodobnostní funkce náhodné veličiny X. Pro $x \in \mathbb{R}$ je

$$p_X(x) = P(X = x) = P(\{\omega \in \Omega | X(\omega) = x\}).$$

Definice 13 (Spojitá náhodná veličina)
Náhodná veličina se nazývá spojitá, pokud existuje nezáporná reálná funkce f_X tak, že pro každé $x \in \mathbb{R}$ je

$$F_X(x) = \int_{-\infty}^{x} f_X(t)dt.$$

V takovém případě se f_X nazývá hustota pravděpodobnosti.

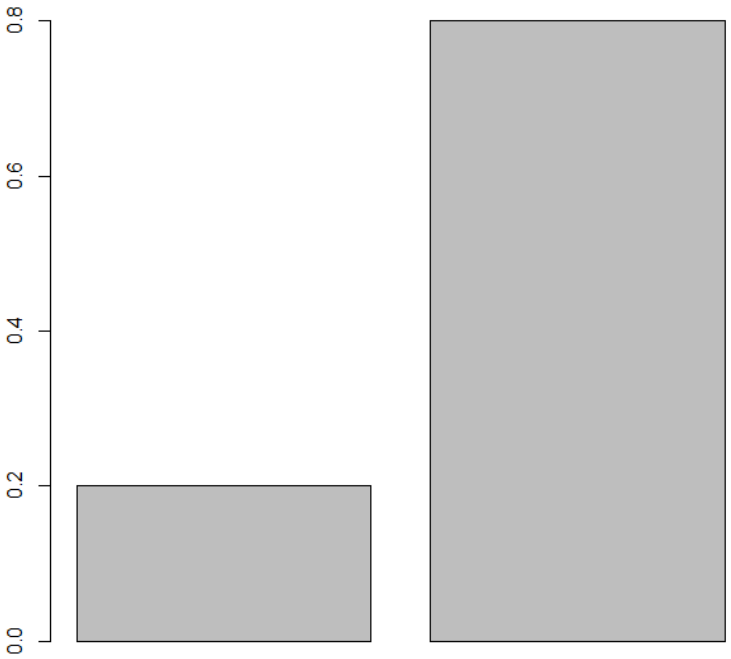
Distribuční funkce náhodné veličiny tvoří rozdělení [13]. Jedním ze základních pravděpodobnostních rozdělení je rovnoměrné rozdělení [14]. To přiřazuje každému elementárnímu jevu stejnou pravděpodobnost.

Dalším pravděpodobnostním rozdělením, které nás bude zajímat, je alternativní (Bernoulliho) rozdělení [15, 16]. Jedná se o rozdělení, ve kterém diskrétní náhodná veličina nabývá hodnoty 1 s pravděpodobností p a 0 s pravděpodobností $1 - p$. Jeho pravděpodobnostní funkce je znázorněna na obrázku 2.

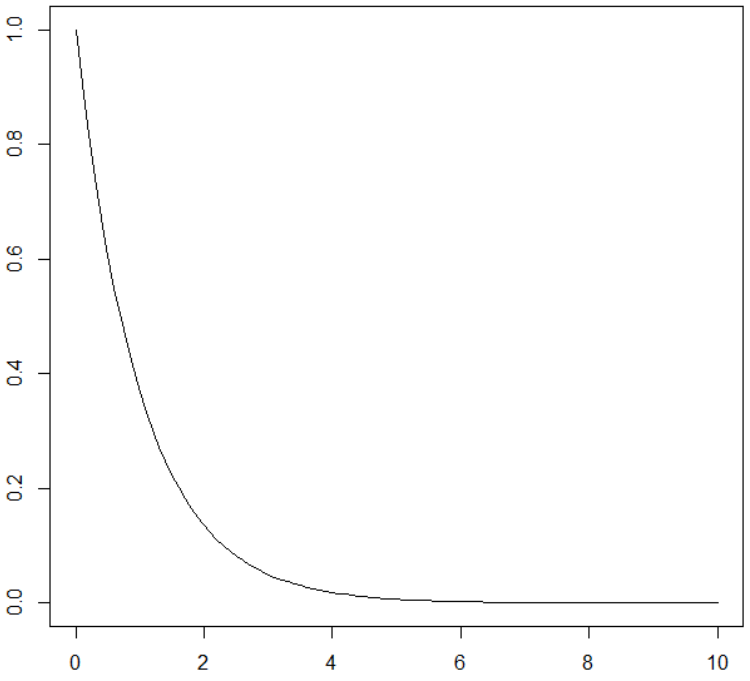
Nakonec si ukážeme ještě exponenciální rozdělení [17] spojité náhodné veličiny. Její hustota funkce je v následujícím tvaru.

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x} & \text{pokud } x > 0, \\ 0 & \text{pokud } x \leq 0. \end{cases}$$

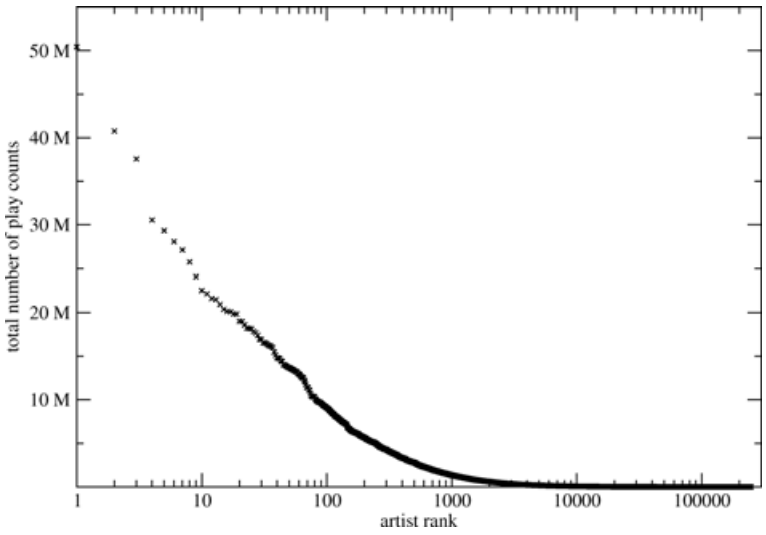
Průběh hustoty exponenciálního rozdělení si můžete prohlédnout na obrázku 3. Více o teorii pravděpodobnosti se dočtete např. v [18].



Obrázek 2: Pravděpodobnostní funkce alternativního rozdělení pro $p = 0,8$.



Obrázek 3: Hustota pravděpodobnosti exponenciálního rozdělení.



Obrázek 4: The Long Tail efekt zachycující popularitu umělců na Last.fm měřenou počtem přehrání. Data byla sbírána v červenci 2007 u 260 525 umělců (pochází z knihy [1]).

4.2 The Long Tail problém

Představme si, že máme obchod s hudbou. Jedná se o zaběhnutý úspěšný obchod, který má velké prodeje a tím pádem i velkou statistiku prodejnosti. Nyní si všechna hudební alba, která prodáváme, setřídíme podle prodejnosti. Zjistíme, že na začátku máme malý počet hodně prodávaných položek, který postupně přejde na velké množství položek s nízkými prodeji. Tohle je právě The Long Tail efekt. Jak vypadá graficky je ilustrováno na obrázku 4.

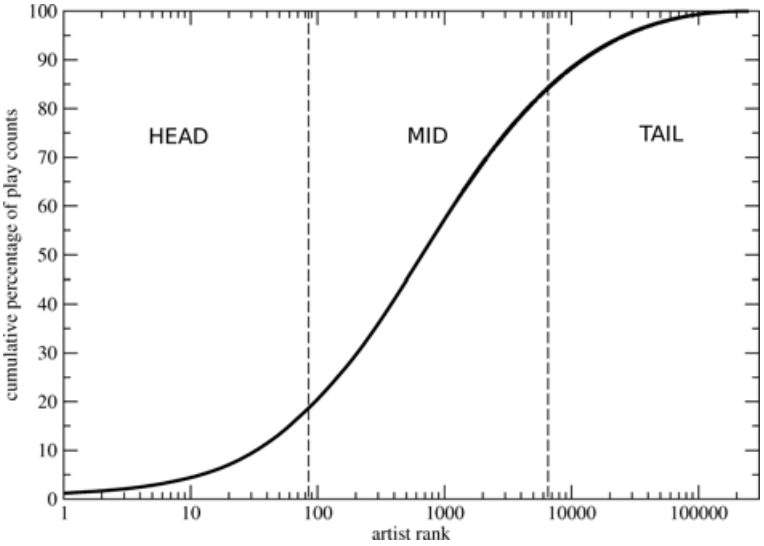
Jak jste si jistě všimli, k práci s The Long Tail potřebujeme nějak změřit popularitu jednotlivých položek. V hudebních online službách se k tomu většinou využívá počet přehrání skladby, alba nebo umělce. Jak tedy matematicky pracovat s popularitou položky? S tím nám pomůže formální definice The Long Tail modelu [19].

Definice 14 (The Long Tail model)

$$F(x) = \frac{\beta}{\left(\frac{N_{50}}{x}\right)^{\alpha} + 1}$$

Kde:

- $F(x)$ – procentní podíl celkového objemu pokrytého objekty do pozice x,
- N_{50} – počet objektů, které pokrývají polovinu celého objemu ($F(N_{50}) = 50$),
- α – faktor, který určuje formu funkce,



Obrázek 5: Kumulativní distribuční funkce na datech z obrázku 4.

- β – celkový objemový podíl.

$F(x)$ je kumulativní distribuční funkce, kterou můžeme rozdělit na tři části: head, mid a tail. Ohraničení mezi head a mid částí je definováno následovně:

$$X_{head \to mid} = N_{50}^{\frac{2}{3}}.$$

A ohraničení mezi mid a tail částí je:

$$X_{mid \to tail} = N_{50}^{\frac{4}{3}} \cong X_{head \to mid}^{2}.$$

Obrázek 5 znázorňuje kumulativní distribuční funkci The Long Tail modelu ilustrovaného na obrázku 4. Všimněte si, že prvních 737 umělců (0,28% všech) tvoří 50% všech přehrání. To znamená $F(737) = 50$. Zároveň $N_{50} = 737$. A rozdělení křivky na tři části probíhá v hodnotách $X_{head \to mid} = 82$ a $X_{mid \to tail} = 6655$.

The Long Tail je pro nás při doporučování hudby důležitý, protože nám ukazuje mainstreamové nálady ve společnosti. Pokud budeme mít uživatele, který poslouchá okrajový nepopulární žánr, je dobré mu doporučit něco podobného a vyvarovat se navrhovaní písniček populárních ve většinové společnosti. Křivka popularity nám také pomůže když doporučujeme novou nebo neznámou hudbu. Pokud budeme doporučovat interprety, alba nebo skladby z tail části, máme o hodně větší jistotu, že ji uživatel nebude už znát.

5 Algoritmus

V této kapitole se konečně dostáváme k návrhu algoritmu. Vybereme data, na kterých budeme doporučování provádět a popíšeme celý algoritmus.

5.1 Hudební databáze

Než začneme algoritmus sestavovat, je potřeba mít nějaká hudební data. Existuje několik databází. My si ukážeme dvě, z nichž si pro účely našeho algoritmu jednu vybereme.

5.1.1 MusicBrainz

MusicBrainz [20, 21, 22] je hudební encyklopedie s volně přístupným obsahem. Původně bylo založeno s cílem nahradit komerční CDDB (Compact Disc Database), databázi pro software, který vyhledává informace o audio CD na internetu. MusicBrainz mimo jiné obsahuje informace o interpretech (životopis, tvorba atd.), albech (seznam skladeb, datum vydání, obrázek alba atd.) i skladbách (název, skladatel, délka stopy atd.). K 26. červenci 2016 databáze obsahovala kolem 1,1 milionů umělců, 1,6 milionů alb a 16 milionů nahrávek.

Jak již bylo řečeno, jedná se o volně dostupnou databázi, takže je možné si ji stáhnout [23].

5.1.2 Discogs

Discogs [24, 25] (vzniklo zkrácením slova discographies) je volně přístupná hudební databáze, kterou vlastní firma Zink Media, Inc. Přestože obsahuje všechny žánry a formáty, je známá především tím, že je největší databází elektronické hudby a vinylových médií. Discogs obsahuje kolem 9 milionů vydaných alb, 5 milionů umělců a 1 milionu vydavatelů.

Pro účely našeho algoritmu a aplikace jsme vybrali právě tuto databázi. Hlavním důvodem je, že obsahuje informace o žánrech a hudebních stylech. MusicBrainz sice tyto informace eviduje také, ale formou štítků. Což je limitující, protože štítky neobsahují čistě informace jen o žánru nebo stylu. Zato Discogs má pole pro tyto údaje s omezenou množinou možností výběru. Data jsme získali z [26] (obraz k září 2017) a zpracovali pomocí scriptu [27], který nám tato data naimportoval do PostgreSQL. Vybírali jsme pouze data s kvalitami: Complete And Correct (úplné a správné), Correct (správné), Needs Minor Changes (vyžaduje menší změny) a Needs Vote (vyžaduje hlasování).

5.2 Další potřebná data

Náš algoritmus bude doporučovat písničky metodou CB. K tomu však potřebujeme data o jednotlivých nahrávkách. Ta nám zpřístupní Spotify. Audio analýzu

dříve poskytovala služba The Echo Nest [28], kterou v březnu 2014 koupilo právě Spotify. Díky tomu získáme audio analýzu skladeb, včetně jejich ukázek.

Jaké informace z audio analýzy [29] Spotify poskytuje si můžete prohlédnout v následujícím přehledu:

- *acousticness* – Hodnota od 0,0 do 1,0 určující akustičnost nahrávky. 1,0 reprezentuje, že je skladba vysoce akustická.
- *danceability* – Hodnota od 0,0 do 1,0 měřící, jak moc je písnička vhodná k tancování. To závisí na hudebních elementech zahrnujících tempo nebo stabilitu rytmu. 1,0 reprezentuje nejvyšší míru tanečnosti.
- *duration_ms* – Délka skladby v milisekundách.
- *energy* – Měří energičnost v hodnotách od 0,0 do 1,0. Typická energická skladba je pocitově rychlá, hlasitá a rušivá.
- *instrumentalness* – Predikuje, zda-li nahrávka obsahuje vokály. Čím víc se hodnota blíží 1,0, tím je instrumentálnější.
- *key* – Určuje tóninu skladby. Řídí se standardem Pitch Class notace. Např. C = 0 nebo D = 2.
- *liveness* – Detekuje přítomnost publika v nahrávce. Čím vyšší hodnota, tím větší je pravděpodobnost, že byla písnička nahrána na živém koncertu.
- *loudness* – Hlasitost skladby v decibelech.
- *mode* – Modalita skladby (1 = majoritní, 0 = minoritní).
- *speechiness* – Detekuje v písničce mluvené slovo (1,0 = mluvené slovo, 0,0 zpívaná písnička).
- *tempo* – Rychlost skladby v beatech za minutu.
- *time_signature* – Vlastnost skladby v čase.
- *valence* – Hodnota od 0,0 do 1,0 měřící pozitivnost skladby. Písničky s vysokou valencí jsou pozitivní (např. šťastné, bláznivé, euforické), zatímco ty s nízkou valencí jsou negativní (např. smutné, depresivní, naštvané).

Spotify nám nepomůže jen s audio analýzou a zvukovými ukázkami. Obsahuje také informace o popularitě skladeb, interpretů a alb. Všechna tato data jsme získali pomocí jednoduchého scriptu, který je stáhl přes Spotify Web API [30].

5.2.1 Vzdálenost skladeb

Pro účely našeho algoritmu potřebujeme, aby měl k dispozici vypočítané vzdálenosti mezi jednotlivými skladbami. K výpočtu vzdálenosti mezi dvěma skladbami jsme využili již dříve zmíněné Manhatonské vzdálenosti. Trochu jsme ji ale upravili. Vytvořili jsme funkci $contains(A, B)$, jejímiž argumenty jsou dvě množiny. Její předpis vypadá následovně:

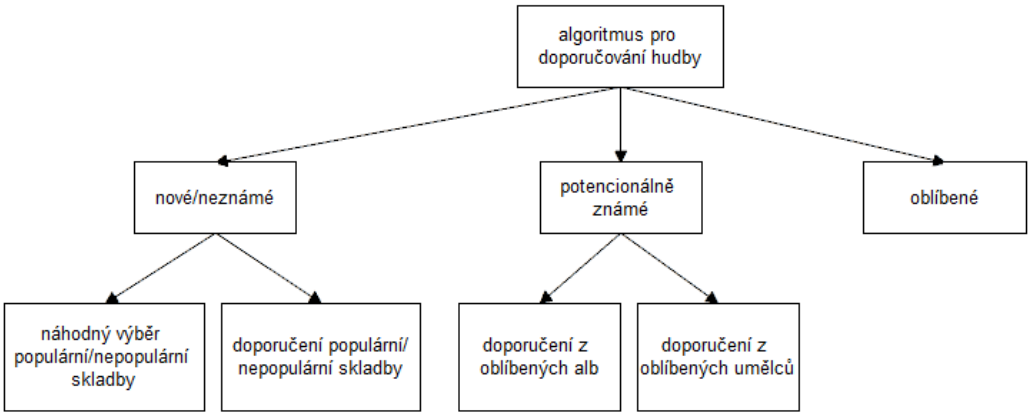
Definice 15 (Funkce $contains(A, B)$)

$$contains(A, B) = \begin{cases} 0 & \text{pokud } A \cap B \neq \emptyset, \\ 1 & \text{pokud } A \cap B = \emptyset. \end{cases}$$

U každé skladby máme k dispozici informace z audio analýzy a z informací o albu také rok vydání, zemi (získali jsme z hlavního vydání alba), hudební žánry a styly. Všechny číselné hodnoty (to znamená všechny kromě země, žánrů a stylů) jsme normalizovali na interval $\langle 0, 1\rangle$. Naše funkce počítající vzdálenost skladeb vypadá následovně:

Definice 16 (Funkce $similarity(i, j)$)

$$\begin{aligned} similarity(i, j) = {} & |i_{acousticness} - j_{acousticness}| + \\ & + |i_{danceability} - j_{danceability}| + \\ & + |i_{duration\_ms} - j_{duration\_ms}| + \\ & + |i_{energy} - j_{energy}| + \\ & + |i_{instrumentalness} - j_{instrumentalness}| + \\ & + |i_{key} - j_{key}| + \\ & + |i_{liveness} - j_{liveness}| + \\ & + |i_{loudness} - j_{loudness}| + \\ & + |i_{mode} - j_{mode}| + \\ & + |i_{speechiness} - j_{speechiness}| + \\ & + |i_{tempo} - j_{tempo}| + \\ & + |i_{time\_signature} - j_{time\_signature}| + \\ & + |i_{valence} - j_{valence}| + \\ & + |i_{year} - j_{year}| + \\ & + contains(\{i_{country}\}, \{j_{country}\}) + \\ & + contains(i_{genres}, j_{genres}) + \\ & + contains(i_{styles}, j_{styles}). \end{aligned}$$



Obrázek 6: Struktura algoritmu.

5.3 Návrh algoritmu

Nyní máme vše potřebné k tomu, abychom mohli sestavit náš algoritmus pro doporučování hudby. Jeho zavolání bude znamenat požadavek na další skladbu. Ten poté vybere jeden z konkrétních modulů, které vrátí navrhovanou písničku. Na obrázku 6 je diagram zobrazující strukturu modulů algoritmu.

Doporučení hudby jsme rozdělili do třech základních modulů. První z nich bude navrhovat nové nebo neznámé skladby. Druhý bude vybírat potencionálně známou hudbu. Za tu jsou považovány skladby, které jsou:

- na albu, na kterém se vyskytuje již oblíbená písnička,
- od interpreta, jehož skladba se uživateli líbila.

Třetí základní modul bude vracet již oblíbenou hudbu.

5.3.1 Výběr nové nebo neznámé skladby

Začneme nejzákladnějším a nejobecnějším modulem. Jeho úkolem je náhodně vybrat skladbu na základě zadaných parametrů, kterými jsou popularita, žánr a desetiletí. Tento modul je velmi důležitý, protože při neúspěchu se na něj budou ostatní odvolávat.

Moduly budou používat dvě pomocné funkce. První je *FilterTracks* (algoritmus 1), která redukuje výběr nahrávek podle zadaného žánru a desetiletí. Druhou je *ClearSetOfTracks* (algoritmus 2), jejímž účelem je očistit vstupní množinu o nežádoucí skladby. Mezi ty řadíme skladby:

- neoblíbené,
- vyskytující se mezi posledními 50 přehranými,

- skladby posledních 20 přehraných interpretů.

Algoritmus 1: FilterTracks

Input: *tracks* – a set of all tracks
genre – a selected genre
decade – a selected decade
Output: a set of filtered tracks

```
1  selected ← ∅;
2  foreach t ∈ tracks do
3      switch decade do
4          case "all" do
5              if genre ∈ t.genre or genre = "all" then
                   selected ← selected ∪ t;
6          case "10s" do
7              if t.year ≥ 2010 and (genre ∈ t.genre or genre = "all") then
                   selected ← selected ∪ t;
8          case "00s" do
9              if t.year ≥ 2000 and t.year < 2010 and (genre ∈ t.genre or
                   genre = "all") then selected ← selected ∪ t;
10         case "90s" do
11             if t.year ≥ 1990 and t.year < 2000 and (genre ∈ t.genre or
                   genre = "all") then selected ← selected ∪ t;
12         case "80s" do
13             if t.year ≥ 1980 and t.year < 1990 and (genre ∈ t.genre or
                   genre = "all") then selected ← selected ∪ t;
14         case "70s" do
15             if t.year ≥ 1970 and t.year < 1980 and (genre ∈ t.genre or
                   genre = "all") then selected ← selected ∪ t;
16         case "60s" do
17             if t.year < 1970 and (genre ∈ t.genre or genre = "all") then
                   selected ← selected ∪ t;
18         end
19     end
20 end
21 return selected
```

Výsledný modul (algoritmus 3) nejprve vyfiltruje množinu skladeb podle žánru a desetiletí a odebere nežádoucí nahrávky, k nimž přidá i oblíbené. Výsledek setřídí podle popularity. Pokud má vrátit populární skladbu, náhodně ji vybere z první desetiny setřízené množiny. V opačném případě ji náhodně vybere ze zbytku.

Proč jsme jako zlom popularity určili jednu desetinu? Je to proto, že data o popularitě, která poskytuje Spotify, už jsou nějakým způsobem vygenerovaná

Algoritmus 2: ClearSetOfTracks

Input: *tracks* – a set of all tracks
artists – a set of all artists
user – a logged in user
Output: a set of tracks, which needs to be removed from recommendation

```
1  union ← ∅;
2  union ← union ∪ user.black_list;
3  history ← select last 50 tracks from user.plays_history;
4  union ← union ∪ history;
5  history ← select last 20 tracks from user.plays_history;
6  art ← ∅;
7  foreach h ∈ history do
8      art ← art ∪ h.artists;
9  end
10 foreach t ∈ tracks do
11     foreach a ∈ art do
12         if a ∈ t.artists then union ← union ∪ t;
13     end
14 end
15 return union
```

a udávaná celočíselně v procentech. Tato data se tedy obtížně analyzují. Přibližně jedna čtvrtina skladeb má dokonce nulovou popularitu. Sečetli jsme tedy tyto hodnoty a získali medián, který se nachází kousek nad jednou desetinou všech dat. Toto číslo jsme zvolili tak, aby populárních skladeb nebylo příliš málo a zároveň příliš mnoho.

5.3.2 Doporučení nové nebo neznámé skladby

Nejdůležitějším modulem celého algoritmu je ten pro doporučení nové nebo neznámé skladby (algoritmus 4). Doporučení provede průchodem oblíbených skladeb, kde ke každé vybere 200 nejbližších. Výslednou množinu vyfiltruje a odebere nežádoucí nahrávky, včetně oblíbených. Pokud nezbude prázdná množina, stejně jako předchozí modul, vrátí výsledek podle popularity. V opačném případě zavolá *ChooseNewOrUnknown*.

5.3.3 Nové nebo neznámé

Tento modul stojí v hierarchii nad moduly pro výběr a doporučení nové nebo neznámé skladby. Jeho účelem je zamezit možnému zastavení přísunu nových písniček. Mohlo by totiž dojít k opakovanému volání *RecommendNewOrUnknown*. To by způsobilo, že označení jedné skladby jako oblíbené by algoritmus vedlo jen jedním směrem (tedy jen ke 200 skladbám, které jsou té oblíbené nejbližší). Tento

Algoritmus 3: ChooseNewOrUnknown

Input: *tracks* – a set of all tracks
artists – a set of all artists
popularity – recommend a popular/unpopular track
genre – a choosed genre
decade – a choosed decade
user – a logged in user

Output: a recommended track

1 $selected \rightarrow FilterTracks(tracks, genre, decade)$;
2 $union \leftarrow ClearSetOfTracks(selected, artists, user)$;
3 $union \leftarrow union \cup user.white\_list$;
4 $selected \leftarrow selected \setminus union$;
5 sort *selected* by a popularity in descending order;
6 **if** *popularity* **then**
7 $result \leftarrow$ select first $\frac{selected.count}{10}$ from *selected*;
8 $track \leftarrow$ random select an item from *result*;
9 **return** *track*
10 **else**
11 $result \leftarrow$ select last $\frac{selected.count*9}{10}$ from *selected*;
12 $track \leftarrow$ random select an item from *result*;
13 **return** *track*
14 **end**

Algoritmus 4: RecommendNewOrUnknown

Input: *tracks* – a set of all tracks
artists – a set of all artists
popularity – recommend a popular/unpopular track
genre – a choosed genre
decade – a choosed decade
user – a logged in user

Output: a recommended track

```
1  selected ← ∅;
2  union ← ClearSetOfTracks(tracks, artists, user);
3  union ← union ∪ user.white_list;
4  foreach w ∈ user.white_list do
5      selected ← selected ∪ w.near_tracks;
6  end
7  selected ← FilterTracks(selected, genre, decade);
8  selected ← selected \ union;
9  if selected ≠ ∅ then
10     sort selected by a popularity in descending order;
11     if popularity then
12         result ← select first selected.count/10 from selected;
13         track ← random select an item from result;
14         return track
15     else
16         result ← select last selected.count*9/10 from selected;
17         track ← random select an item from result;
18         return track
19     end
20 else
21     return
         ChooseNewOrUnknown(tracks, artists, popular, genre, decade, user)
22 end
```

modul (reprezentovaný algoritmem 6) bude přibližně po každém pátém volání volat pro oživení doporučení *ChooseNewOrUnknown*. Docílí toho díky generování náhodných čísel s alternativním (Bernoulliho) rozdělením, která jednoduše získáme z rovnoměrného rozdělení (viz algoritmus 5). Právě tomuto rozdělení se totiž většinou generátory pseudonáhodných čísel blíží (viz [31, 32]).

Algoritmus 5: BernoulliDistribution

Input: p – a real number from $\langle 0, 1\rangle$
Output: a random non-negative integer with Bernoulli distribution

```
1 r ← a random real number from ⟨0, 1⟩;
2 if r < p then
3     return 1
4 else
5     return 0
6 end
```

Algoritmus 6: NewOrUnknown

Input: *tracks* – a set of all tracks
artists – a set of all artists
popularity – recommend a popular/unpopular track
genre – a choosed genre
decade – a choosed decade
user – a logged in user
Output: a recommended track

```
1 if BernoulliDistribution(0, 8) = 1 then
2     return
        RecommendNewOrUnknown(tracks, artists, popularity, genre, decade,
        user)
3 else
4     return
        ChooseNewOrUnknown(tracks, artists, popularity, genre, decade, user)
5 end
```

5.3.4 Doporučení z oblíbených alb

Algoritmus 7 má na starosti doporučení prostřednictvím oblíbených alb. Jak již bylo dříve zmíněno, za oblíbené album považujeme to, na kterém se vyskytuje oblíbená skladba. Algoritmus proto projde všechny oblíbené písničky a do výsledku zařadí obsah všech alb, na kterých se nacházejí. Pak už jen tuto množinu vyfiltruje funkcí *FilterTracks*, očistí *ClearSetOfTracks* a odebere oblíbené skladby. Pokud není výsledná množina prázdná, náhodně z ní vybere výslednou skladbu. V opačném případě zavolá modul *NewOrUnknown*.

Algoritmus 7: FavoriteAlbums

Input: *tracks* – a set of all tracks
artists – a set of all artists
genre – a choosed genre
decade – a choosed decade
user – a logged in user

Output: a recommended track

```
1  union ← ClearSetOfTracks(tracks, artists, user);
2  union ← union ∪ user.white_list;
3  selected ← ∅;
4  alb ← ∅;
5  foreach t ∈ user.white_list do
6      alb ← alb ∪ t.album;
7  end
8  foreach a ∈ alb do
9      selected ← selected ∪ a.tracks;
10 end
11 selected ← FilterTracks(selected, genre, decade);
12 selected ← selected \ union;
13 if selected ≠ ∅ then
14     track ← random select an item from selected;
15     return track
16 else
17     return NewOrUnknown(tracks, artists, true, genre, decade, user)
18 end
```

5.3.5 Doporučení z oblíbených umělců

Mezi potencionálně známými budeme doporučovat i písničky oblíbených umělců. To ilustruje algoritmus 8, který z oblíbených skladeb shromáždí jejich interprety. Najde všechny jejich skladby a jako předchozí moduly vyfiltruje a odebere nežádoucí skladby (mezi které patří také oblíbené). Stejně jako u předchozího algoritmu, pokud zůstane neprázdná množina, vrátí náhodně vybranou položku. Pokud ne, zavolá modul *NewOrUnknown*.

Algoritmus 8: FavoriteArtists

Input: $tracks$ – a set of all tracks
$artists$ – a set of all artists
$genre$ – a choosed genre
$decade$ – a choosed decade
$user$ – a logged in user

Output: a recommended track

```
1  union ← ClearSetOfTracks(tracks, artists, user);
2  union ← union ∪ user.white_list;
3  selected ← ∅;
4  art ← ∅;
5  foreach t ∈ user.white_list do
6      art ← art ∪ t.artists;
7  end
8  foreach a ∈ art do
9      selected ← selected ∪ a.tracks;
10 end
11 selected ← FilterTracks(selected, genre, decade);
12 selected ← selected \ union;
13 if selected ≠ ∅ then
14     track ← random select an item from selected;
15     return track
16 else
17     return NewOrUnknown(tracks, artists, true, genre, decade, user)
18 end
```

5.3.6 Doporučení potencionálně známé skladby

Potencionálně známou skladbu snadno doporučíme díky již dříve zmíněným modulům. Ty budeme volat náhodně (viz algoritmus 9) s rovnoměrným rozdělením pravděpodobnosti.

Algoritmus 9: PotencionalKnown

Input: *tracks* – a set of all tracks
artists – a set of all artists
genre – a choosed genre
decade – a choosed decade
user – a logged in user

Output: a recommended track

```
r ← generate a random natural number from ⟨0, 1⟩;
if r = 0 then
    return FavoriteArtists(tracks, artists, genre, decade, user)
else
    return FavoriteAlbums(tracks, artists, genre, decade, user)
end
```

5.3.7 Oblíbené skladby

Modul pro vrácení oblíbených skladeb (algoritmus 10) jednoduše vyfiltruje všechny oblíbené podle zadaných parametrů, odebere nežádoucí a z výsledku náhodně vybere jednu položku. Pokud je výsledek prázdný, zavolá *PotencionalKnown*.

Algoritmus 10: Favorites

Input: *tracks* – a set of all tracks
artists – a set of all artists
genre – a choosed genre
decade – a choosed decade
user – a logged in user

Output: a favorite track

```
selected ← FilterTracks(user.white_list, genre, decade);
union ← ClearSetOfTracks(tracks, artists, user);
selected ← selected \ union;
if selected ≠ ∅ then
    track ← random select an item from selected;
    return track
else
    return PotencionalKnown(tracks, artists, genre, decade, user)
end
```

5.3.8 Doporučení skladby

Nyní se dostáváme k výslednému algoritmu 15. V něm budeme používat funkce pro generování náhodných čísel s alternativním (Bernoulliho) a exponenciálním pravděpodobnostním rozdělením. K vygenerování exponenciálního rozdělení vy-

užijeme inverzní funkce [33, 34], která rovnoměrné rozdělení převede na námi požadované (algoritmus 11).

Algoritmus 11: ExponencialDistribution

Input: τ
Output: a random non-negative integer with exponencial distribution

$\lambda \leftarrow 1$;
$r \leftarrow$ a random real number from $\langle 0, 1 \rangle$;
return *round* $\frac{-\log(1-(1-e^{-\lambda * \tau})) * r}{\lambda}$ *to an integer*

Nakonec budeme ještě potřebovat funkci, která spočítá index popularity. Tento index bude určovat, do jaké míry uživatel poslouchá populární a nepopulární hudbu. Jeho výpočet ilustruje algoritmus 12. Index se bude počítat jako vážený průměr popularity oblíbených interpretů, kde váha je počet výskytů interpreta mezi oblíbenými písničkami. Pro jednoduchost považujeme interprety s popularitou nad 50% za populární a naopak pod 50% za nepopulární. Navíc toto procento rozděluje umělce opět poblíž jedné desetiny.

Algoritmus 12: GetIndexOfPopularity

Input: *user* – a logged in user
Output: an index of popularity

let *art* be a dictionary;
foreach $t \in user.white\_list$ **do**
    foreach $a \in t.artists$ **do**
        if $a \in art$ **then**
            $art(a) \leftarrow art(a) + 1$;
        else
            $art(a) \leftarrow 0$;
        end
    end
end
$sum \leftarrow 0$;
foreach $(key, value) \in art$ **do**
    $sum \leftarrow sum + key.popularity * value$;
end
$index \leftarrow \frac{sum}{art.lenght}$;
return $\frac{index}{100}$

Doporučení algoritmu bude možné ovlivnit třemi parametry: módem, žánrem a desetiletím. Proč je v algoritmu využíváme? Podle studie *Phoenix 2 UK Project* [1] z roku 2006, která zkoumala populaci mezi 16 a 45 lety, existuje několik stupňů zájmu lidí o hudbu (typů posluchačů):

1. Znalci (savants): Zdá se, že všechno v jejich životě je provázáno s hudbou. Jejich hudební znalosti jsou velmi rozsáhlé. Pro tento typ lidí je doporučování hudby nejtěžší, protože jsou velmi nároční. Celkem tvoří 7% zkoumané skupiny.

2. Nadšenci (enthusiasts): Hudba je jejich klíčovou součástí života, ale je vyvažována i jinými zájmy. Tito lidé tvoří 21% zkoumané skupiny.

3. Příležitostní (casuals): Hudba je v jejich životě vítaná, ale jiné zájmy jsou pro ně mnohem důležitější. Ze zkoumané skupiny tvoří 32%.

4. Lhostejní (indifferents): Jejich spánek by příliš neovlivnilo, kdyby hudba neexistovala. Reprezentují 40% zkoumané skupiny.

Znalci nepotřebují populární, ale riskantní a chytrá doporučení. Nadšenci balancují mezi zajímavým a neznámým, nebo známým doporučením. Příležitostným a lhostejným posluchačům stačí doporučovat populární a mainstreamové písničky. Náš algoritmus nebude tyto typy uživatelů detekovat, ale dá jim možnost si zvolit ze tří různých módů doporučení:

1. Známé (algoritmus 13): Doporučí populární neznámé, potencionálně známé nebo oblíbené skladby.

2. Automat: Na základě indexu popularity bude vybírat mezi prvním a třetím módem.

3. Průzkum (algoritmus 14): Doporučí nepopulární neznámé nebo oblíbené skladby.

Našim cílem je také zohlednit aktuální náladu posluchače. Toho dosáhneme tím, že mu dáme na výběr ze žánrů a desetiletí. Výsledné doporučení algoritmu 15 tedy bude zohledňovat typ posluchače, ale také jeho náladu.

Algoritmus 13: ModeOne

Input: *tracks* – a set of all tracks
artists – a set of all artists
genre – a choosed genre
decade – a choosed decade
user – a logged in user

Output: a recommended track

```
1 switch ExponencialDistribution(3) do
2     case 0 do
3         return
            NewOrUnknown(tracks, artists, true, genre, decade, user)
4     case 1 do
5         return PotencionalKnown(tracks, artists, genre, decade, user)
6     case 2 do
7         return Favorites(tracks, artists, genre, decade, user)
8     end
9 end
```

Algoritmus 14: ModeThree

Input: *tracks* – a set of all tracks
artists – a set of all artists
genre – a choosed genre
decade – a choosed decade
user – a logged in user

Output: a recommended track

```
1 switch ExponencialDistribution(3) do
2     case 0 do
3         return
            NewOrUnknown(tracks, artists, false, genre, decade, user)
4     case 1 do
5         return
            NewOrUnknown(tracks, artists, false, genre, decade, user)
6     case 2 do
7         return Favorites(tracks, artists, genre, decade, user)
8     end
9 end
```

Algoritmus 15: RecommendNextSong

Input: *tracks* – a set of all tracks
artists – a set of all artists
mode – a mode of recommendation
genre – a choosed genre
decade – a choosed decade
user – a logged in user

Output: a recommended track

```
1  switch mode do
2      case 1 do
3          return ModeOne(tracks, artists, genre, decade, user)
4      case 2 do
5          if BernouliDistribution(GetIndexOfPopularity(user)) = 1 then
6              return ModeOne(tracks, artists, genre, decade, user)
7          else
8              return ModeThree(tracks, artists, genre, decade, user)
9          end
10     case 3 do
11         return ModeThree(tracks, artists, genre, decade, user)
12     end
13 end
```

6 Implementace

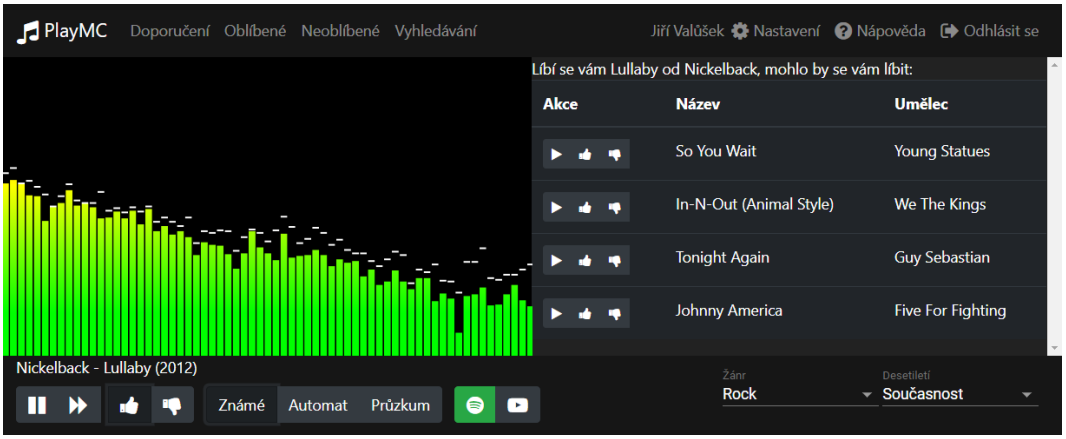
Navržený algoritmus jsme implementovali formou webové aplikace. Po registraci a přihlášení si uživatel může vybrat ze tří módů v kombinaci s možností volby žánru a desetiletí. Na základě těchto parametrů jsou přehrávány skladby. Posluchač může vyvíjet zpětnou vazbu pomocí tlačítek pro označení hrající písničky jako to se mi líbí, nebo to se mi nelíbí. Nechybí ani možnost vybrat mezi přehráváním 30 sekundových ukázek ze Spotify, nebo přehrávačem Youtube.

Pro urychlení adaptace algoritmu na daného uživatele jsou mimo přehrávané písničky zobrazena i další 4 doporučení. Je také možné si oblíbenou písničku aktivně vyhledat v databázi. Vše doplňuje možnost prohlédnou si (případně přehrát) oblíbené i neoblíbené skladby.

Z technického hlediska je aplikace napsána v jazyce PHP [35], JavaScript [36], HTML [37] a CSS [38]. Využívá šablonovací systém Smarty [39], grafickou knihovnu Bootstrap [40], Font Awesome [41] a technologii Polymer [42]. Pro vizualizaci audio spektra (obrázek 7) využívá upraveného kódu [43]. Obrázky byly získány z webu Pixabay [44]. Data jsou uložena v PostgreSQL [45] databázi. Jak bylo již dříve zmíněno, aplikace pracuje s daty z Discogs a Spotify. Ta po sloučení těchto dvou zdrojů a následné selekci tvoří 84 068 umělců, 121 079 alb a 980 460 skladeb.

Na následujícím příkladu si ukážeme, jak algoritmus doporučuje. Přihlásili jsme se jako uživatel, který už tuto aplikaci nějakou dobu používá. Nastavili jsme žánr na pop, desetiletí na současnost (písničky od roku 2010) a mód do režimu, který bude vracet známé skladby. Následně jsme dostávali doporučení v tomto pořadí:

1. Kovacs - Wolf In Cheap Clothes,
2. Kylie Minogue - Get Outta My Way,
3. Rachel Platten - Congratulations,
4. Kat Dahlia - Clocks,
5. Revolverheld - Ich Lass Für Dich Das Licht An,
6. Calvin Harris - Pray To God,
7. A Great Big World - Everyone Is Gay,
8. Avril Lavigne - Let Me Go,
9. Haloo Helsinki! - Rakkaus,
10. Rasmus Seebach - Uanset.



Obrázek 7: Náhled aplikace.

Ze znalosti oblíbených skladeb v uživatelském profilu se dá určit, že algoritmus vrátil 2 oblíbené skladby (Calvin Harris - Pray To God, Avril Lavigne - Let Me Go), 1 potencionálně známou (protože mezi oblíbenými je písnička od Kylie Minogue) a zbylých 7 pravděpodobně vrátil modul pro nové nebo neznámé skladby.

Webová aplikace je k dispozici na adrese https://tux.inf.upol.cz/~valuji00/.

6.1 Návrhy na vylepšení

Algoritmus využívá metody CB, což nezohledňuje sociální vazby. Tento nedostatek by vyřešilo zkombinování současně používané techniky s CF, případně i využití umělé inteligence.

Když uživatel poslouchá doporučované skladby, nemusí se včas dozvědět, že jeho oblíbený umělec vydal novou písničku. Tuto funkci by mohlo vyřešit přidání dalšího modulu.

Dalším možným vylepšením algoritmu je přidání dynamického výpočtu vzdáleností mezi skladbami. Mělo by být možné přidat novou skladbu do systému, aniž by se vše muselo přepočítávat. Změnou parametrů se mění i křivka popularity. Algoritmus by mohl zlom mezi popularitou a nepopularitou počítat rovněž dynamicky.

Závěr

Nejprve jsme se obecně podívali na problematiku doporučovacích systémů. Konkrétně jsme definovali, co to vlastně je problém doporučení. Následně jsme si uvedli metody, které se k řešení tohoto problému používají. Byly to Demographic Filtering, Collaborative Filtering, Content-Based Filtering, Context-Based Filtering a Hybridní metoda. Dále jsme se zaměřili na problematiku doporučování hudby, včetně příkladu služeb ze současnosti, které jej realizují.

K pochopení námi navrženého algoritmu je potřeba znát některé pojmy z teorie pravděpodobnosti a problematiku The Long Tail efektu, čemuž jsme se věnovali ve čtvrté kapitole.

Navrhli jsme algoritmus pro doporučování hudby, který se ke každému uživateli chová individuálně (za předpokladu, že mu uživatel dává zpětnou vazbu). Snaží se zohledňovat typ a náladu posluchače. Algoritmus rovněž zamezuje častému opakování skladeb a umělců.

Celý algoritmus jsme implementovali formou webové aplikace, která přináší i další funkce. Mezi ně patří možnost vyhledání oblíbené nahrávky nebo doporučení čtyřech dalších skladeb.

Conclusions

At first we introduced recommender systems in general. We defined a recommendation problem and sumarised technics of solutions, namely Demographic Filtering, Collaborative Filtering, Content-Based Filtering, Context-Based Filtering and Hybrid. Then we focused on a music recommendation problem and nowdays services which realise the music recommendation.

For a good understunding of the algorithm we need some prerequisites from probability theory and The Long Tail effect, which were presented in chapter 4.

We created the music recommendation algorithm which behaves individually to users. It tries to take into account type and mood of a listener. The algorithm prevents frequent repeat tracks and artists.

We implemented the algorithm as a web application, which has some additional features. There is a function for searching tracks, or a recommendation of next four tracks.

A Obsah přiloženého CD/DVD

`doc/`
: Text práce ve formátu PDF.

`src/`
: Kompletní zdrojové kódy aplikace. Rovněž obsahuje vše potřebné k jejímu spuštění.

`readme.txt`
: Instrukce pro spuštění aplikace.

Literatura

[1] CELMA, Oscar. *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. 2010. ISBN 978-3-642-13286-5.

[2] *Matrix Factorization Model in Collaborative Filtering Algorithms: A Survey*. [online]. [cit. 2018-3-16]. Dostupný z: ⟨https://www.sciencedirect.com/science/article/pii/S1877050915007462⟩.

[3] C. KAUFMAN, Jaime. *A Hybrid Approach to Music Recommendation: Exploiting Collaborative Music Tags and Acoustic Features*. 2014.

[4] *Spotify*. [online]. [cit. 2018-3-14]. Dostupný z: ⟨https://www.spotify.com/cz/⟩.

[5] VÁCLAVÍK, Lukáš. *Spotify má 71 milionů předplatitelů, ale prodělalo už přes 2 miliardy eur* [online]. [cit. 2018-3-14]. Dostupný z: ⟨https://www.cnews.cz/spotify-71-milionu-predplatitelu⟩.

[6] UNDERWOOD, Corinna. *Use Cases of Recommendation Systems in Business – Current Applications and Methods* [online]. [cit. 2018-3-14]. Dostupný z: ⟨https://www.techemergence.com/use-cases-recommendation-systems/⟩.

[7] URBAN, Petr. *Streamování hudby táhne. Apple Music si předplácí už 38 milionů lidí* [online]. [cit. 2018-3-14]. Dostupný z: ⟨https://www.cnews.cz/apple-music-38-milionu-predplatitelu/⟩.

[8] *Apple Music*. [online]. [cit. 2018-3-14]. Dostupný z: ⟨https://www.apple.com/cz/music/⟩.

[9] *Youtube*. [online]. [cit. 2018-3-14]. Dostupný z: ⟨https://www.youtube.com⟩.

[10] *Google Brain*. [online]. [cit. 2018-3-14]. Dostupný z: ⟨https://research.google.com/teams/brain/⟩.

[11] *Last.fm*. [online]. [cit. 2018-3-15]. Dostupný z: ⟨https://www.last.fm⟩.

[12] *Last.fm – Music Recommendation incorporating social network ties and collaborative filtering*. [online]. [cit. 2018-3-15]. Dostupný z: ⟨http://blogs.cornell.edu/info2040/2012/09/20/last-fm-music-reccomendation-incorporating-social-network-ties-and-collaborative-filtering/⟩.

[13] *Wikipedie – Distribuční funkce*. [online]. [cit. 2018-4-27]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Distribu%C4%8Dn%C3%AD_funkce⟩.

[14] *Wikipedie – Rovnoměrné rozdělení*. [online]. [cit. 2018-4-27]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Rovnom%C4%9Brn%C3%A9_rozd%C4%9Blen%C3%AD⟩.

[15] *Wikipedie – Alternativní rozdělení*. [online]. [cit. 2018-4-27]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Alternativn%C3%AD_rozd%C4%9Blen%C3%AD⟩.

[16] *Wikipedia – Bernoulli distribution*. [online]. [cit. 2018-4-27]. Dostupný z: ⟨https://en.wikipedia.org/wiki/Bernoulli_distribution⟩.

[17] *Wikipedie – Exponenciální rozdělení*. [online]. [cit. 2018-4-27]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Exponenci%C3%A1ln%C3%AD_rozd%C4%9Blen%C3%AD⟩.

[18] ŠTĚPÁN, Karel Zvára – Josef. *Pravděpodobnost a matematická statistika*. 5. vydání. 2012. ISBN 978-80-7378-218-4.

[19] KILKKI, Kalevi. *A practical model for analyzing long tails* [online]. [cit. 2018-3-20]. Dostupný z: ⟨http://ojphi.org/ojs/index.php/fm/article/view/1832/1716⟩.

[20] *MusicBrainz*. [online]. [cit. 2018-3-21]. Dostupný z: ⟨https://musicbrainz.org/⟩.

[21] *Wikipedie – MusicBrainz*. [online]. [cit. 2018-3-21]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/MusicBrainz⟩.

[22] *Wikipedia – MusicBrainz*. [online]. [cit. 2018-3-21]. Dostupný z: ⟨https://en.wikipedia.org/wiki/MusicBrainz⟩.

[23] *MusicBrainz Database / Download*. [online]. [cit. 2018-3-21]. Dostupný z: ⟨https://musicbrainz.org/doc/MusicBrainz_Database/Download⟩.

[24] *Discogs*. [online]. [cit. 2018-3-21]. Dostupný z: ⟨https://www.discogs.com/⟩.

[25] *Wikipedia – Discogs*. [online]. [cit. 2018-3-21]. Dostupný z: ⟨https://en.wikipedia.org/wiki/Discogs⟩.

[26] *Discogs Data*. [online]. [cit. 2018-3-21]. Dostupný z: ⟨http://data.discogs.com⟩.

[27] *Imports the discogs.com monthly XML dumps into databases*. [online]. [cit. 2018-3-21]. Dostupný z: ⟨https://github.com/philipmat/discogs-xml2db⟩.

[28] *Wikipedia – The Echo Nest*. [online]. [cit. 2018-3-21]. Dostupný z: ⟨https://en.wikipedia.org/wiki/The_Echo_Nest⟩.

[29] *Web API Object Model*. [online]. [cit. 2018-3-21]. Dostupný z: ⟨https://developer.spotify.com/web-api/object-model/#audio-features-object⟩.

[30] *Spotify Web API*. [online]. [cit. 2018-4-8]. Dostupný z: ⟨https://developer.spotify.com/web-api/⟩.

[31] *Wikipedie – Pseudonáhodná čísla*. [online]. [cit. 2018-4-3]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Pseudon%C3%A1hodn%C3%A1_%C4%8D%C3%ADsla⟩.

[32] *Wikipedie – Generátor pseudonáhodných čísel*. [online]. [cit. 2018-4-3]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Gener%C3%A1tor_pseudon%C3%A1hodn%C3%BDch_%C4%8D%C3%ADsel⟩.

[33] *How to sample from an exponential distribution using rejection sampling in PHP*. [online]. [cit. 2018-4-4]. Dostupný z: ⟨https://stats.stackexchange.com/questions/68274/how-to-sample-from-an-exponential-distribution-using-rejection-sampling-in-php⟩.

[34] *Inverse transform sampling*. [online]. [cit. 2018-4-4]. Dostupný z: ⟨https://en.wikipedia.org/wiki/Inverse_transform_sampling⟩.

[35] *PHP*. [online]. [cit. 2018-4-24]. Dostupný z: ⟨http://php.net⟩.

[36] *JavaScript Tutorial*. [online]. [cit. 2018-4-24]. Dostupný z: ⟨https://www.w3schools.com/js/default.asp⟩.

[37] *HTML5 Introduction*. [online]. [cit. 2018-4-24]. Dostupný z: ⟨https://www.w3schools.com/html/html5_intro.asp⟩.

[38] *CSS Tutorial*. [online]. [cit. 2018-4-24]. Dostupný z: ⟨https://www.w3schools.com/css/⟩.

[39] *Smarty: PHP Template Engine*. [online]. [cit. 2018-4-24]. Dostupný z: ⟨https://www.smarty.net⟩.

[40] *Bootstrap · The most popular HTML, CSS, and JS library in the world*. [online]. [cit. 2018-4-24]. Dostupný z: ⟨https://getbootstrap.com⟩.

[41] *Font Awesome*. [online]. [cit. 2018-4-30]. Dostupný z: ⟨https://fontawesome.com/?from=io⟩.

[42] *Polymer Project*. [online]. [cit. 2018-4-24]. Dostupný z: ⟨https://www.polymer-project.org⟩.

[43] *HTML5 Audio Visualizer*. [online]. [cit. 2018-4-8]. Dostupný z: ⟨https://github.com/wayou/html5_audio_visualizer⟩.

[44] *Pixabay*. [online]. [cit. 2018-4-30]. Dostupný z: ⟨https://pixabay.com/cs/⟩.

[45] *PostgreSQL: The world's most advanced open source database*. [online]. [cit. 2018-4-29]. Dostupný z: ⟨https://www.postgresql.org⟩.