Hľadanie podobností v textoch ľudových piesní

Autor: Michal Mižák
Vedúci práce: doc. RNDr. Stanislav Krajči, PhD.

Univerzita Pavla Jozefa Šafárika v Košiciach

Abstrakt Aj keď folklór v posledných rokoch zažíva menšiu renesanciu, ľudová pieseň sa dlhé roky vytrácala z bežného života a ľudia už nevedia texty piesní tak, ako kedysi. Aj preto je našou povinnosťou toto dedičstvo vo forme folklóru a spevu zmysluplne uchovať a starať sa oň. Táto práca sa teda zaoberá dizajnom a vývojom softvéru určeného na vylepšenie a dlhodobú údržbu databázy textov ľudových piesní. Budeme sa o to snažiť nepriamou formou, a to hľadaním podobností týchto textov.

1 Úvod

Inšpiráciou pre túto prácu bola pre nás hudba a folklór. Ľudia, ktorých zaujíma, baví či priam fascinuje folkór tvorili a stále tvoria množstvo dokumentov, kde zaznamenávajú texty rôznorodých ľudových piesní. Tieto texty piesní sú takmer vždy zaznamenávané z počutia a, ako si asi všetci vieme predstaviť, nie každý ovláda správny zápis ľudovej reči, rovnako ako každý má tendenciu text pretvoriť podľa toho, ako si ho zapamätá. Dokumenty sa tým pádom líšia od zapisovateľa k zapisovateľovi a teda rovnaké piesne sú zapísané rôznymi nekorektnými spôsobmi. Ako bonus sa môže stať, že aj keď sú tieto piesne zapísané správne, autenticky, piesne sa históriou prenášali medzi regiónmi a vplyvom nárečia sa texty čiastočne pretvorili podľa miestnych zvyklostí a jazyka. Takéto chyby/ regionálne rozdiely sú zvyčajne v zopár slovách a vždy je jasné, že ide o rovnakú pieseň. Cieľom je teda vytvoriť softvér, ktorý bude takéto rozdiely vedieť nájsť a identifikovať, štatisticky ich vyhodnotiť a vytvoriť zoznam najpodobnejších dokumentov v databáze pre každý zadaný text či dokument.

1.1 Ilustračný príklad

Vezmime 3 útržky textu ľudovej piesne *„Dzivče počarovne“* a označme ich postupne (1), (2), (3):

„Dzivče počarovne nezaľub še do mne“,
„Dzifče počarovne ňezaľup še do mne“,
„Dzifče počaromne ňezaľub śe do mňe“.

Všetko to sú útržky, ktoré sa môžu bežne vyskytnúť v dokumentoch na internete alebo v iných zbierkach piesní. Zväčša sú to zmiešaniny šarišskej a zemplínskej alternatívy tejto piesne. Cieľom je do programu zadať (bez ujmy na všeobecnosti) dokument (1) a vidieť výpis (2) a (3) ako dokumenty podobné (1).

1.2 Existujúce riešenia

Napriek tomu, aká unikátna sa táto práca môže zdať, existuje veľa zdrojov, odkiaľ sa dá čerpať. Najlepšou paralelou je dokonca vedecká oblasť *Information retrieval*, ktorá sa zaoberá hľadaním relevantnej informácie z obrovského množstva dokumentov na základe dopytu. Existuje množstvo nástrojov a knižníc vytvorených práve za týmto účelom, väčšina je ale generických pre každý typ dokumentu. To by nám nedovolilo využiť zaujímavé unikátne vlastnosti štruktúry textu folklórnej piesne. Rozhodli sa teda pre vlastné riešenie, s ktorým budeme vedieť zjednodušiť stredobod záujmu tejto vedeckej oblasti na konkrétny, menší a jednoduchší problém a tým pádom získať presnejšie výsledky ako pri generickom prístupe. Konkrétne metódy z tejto vedeckej oblasti budeme popisovať ďalej v článku.

2 Postup práce

Tvorbu bakalárskej práce sme rozdelili na viac fáz:

1. Voľba jazyka a technológií
2. Príprava databázy
3. Základné porovnávanie
4. Pokročilé porovnávanie
5. Tvorba grafického rozhrania

Týmto fázam sa budeme venovať v nasledujúcich kapitolách.

3 Voľba jazykov a technológií

3.1 Java

Asi najdôležitejšou voľbou na začiatku programovania veľkého softvéru je práve voľba hlavného, nosného programovacieho jazyka. Na škole sme mali viac kurzov, z ktorých sme nabrali skúsenosti v používaní viacerých jazykov, najviac skúseností však máme s jazykom Java. Pre tento jazyk sme sa nerozhodli len kvôli našim schopnostiam, ale aj z iných, relevantnejších dôvodov. Cieľ nášho dizajnu je, aby bolo jednoduché algoritmus rozdeliť na fázy a týmto fázam nezávisle meniť implementáciu. Pri správnom OOP návrhu nám práve toto Java vďaka jej dizajnu ponúkne.

3.2 SQL alebo NoSQL?

Toto bola otázka, na ktorú sme zo začiatku mali jasnú odpoveď: NoSQL v jeho najjednoduchšej podobe - vlastná, nami navrhnutá štruktúra .json súborov. A to preto, lebo sme sa SQL báli ako takzvaného "kanónu na vrabce". Avšak, napriek mnohým výhodám NoSQL nás presvedčila konzistentnosť, overenosť a spoľahlivosť SQL technológií. Prečo? Na začiatku sme boli toho názoru, že naše

dáta budú takmer statické a pri spustení algoritmu sa bude meniť len zopár vstupov v databáze. Po hlbšom naštudovaní problematiky nám ale bolo jasné, že takto to nebude a prevažná časť databázy sa bude meniť zakaždým, čo sa algoritmus spustí a dokonca sa to môže zopakovať viacnásobne počas behu algoritmu. Už len toto samotné by stačilo, aby nás to presvedčilo, nehovoriac o iných problémoch, ako napr. paralelné pripojenia používateľov na webové rozhranie.

4 Príprava databázy

Dokumenty s textami piesní žiaľbohu nie sú nikde centralizované, treba ich teda hľadať a zháňať vlastnými cestami a chodníčkami. Nám sa to podarilo z rôznych zdrojov, počnúc primášom Ľudovej hudby FS Zemplín až po rôzne dokumenty voľne nájdené na internete.

4.1 Naplnenie objektov dátami

Dokumenty s textami bolo nutné zoštrukturalizovať, nakoľko v jednom dokumente sa nachádzalo mnoho piesní a nebolo to strojovo čitateľné. Tieto dáta som čítal Javou a naplnil nimi inštančné premenné objektu *Song*, konkrétne *title* a *lyrics*. V rôznych zdrojoch boli zaznamenané aj iné vlastnosti piesní, ako napríklad *region*, *songStyle* a iné hudobné atribúty, ktoré sa môžu hodiť odborníkom pri analýze regionálnych a iných podobností. Tieto sa neskôr serializujú do SQL databázy.

4.2 Úprava špeciálnych znakov

Tak, ako každý zapíše slová piesne svojím spôsobom, tak aj výskyt špeciálnych znakov je rôznorodý. Pre pracovné účely sme si texty piesní upravili inak, ako boli pôvodne zapísané. Samozrejme, pôvodný text sme ale zachovali. Pozrime sa postupne na rôzne znaky a úpravy, ktoré sme s nimi vykonali.

Opakovacie znamienka Existuje mnoho spôsobov, ako vyznačiť opakovanie útržku textu a viac z nich je korektných. Rozhodli sme sa, že namiesto zjednocovania opakovacích znamienok na jeden typ alebo iného ošetrovania opakovaní je pre náš účel vhodné tieto opakovacie znamienka z textov úplne vymazať a zanedbať ich, pretože nás bude hlavne zaujímať to, či sa dané slovo v texte nachádza alebo nie. Inak povedané, pri našom porovnávacom algoritme početnosť slova v texte je pre pieseň málo charakteristická. Rozhodli sme sa tak aj po zohľadnení toho, že takýchto výkyvov je v databáze málo a teda v konečnom dôsledku je to pre nás málo zaujímavé.

Tri bodky Sem tam sa v texte nájdu tri bodky *"..."* kvôli tomu, že niekedy sa stane, že refrén je rovnaký pre celú pieseň a teda v druhom a ďalších nasledujúcich sa namiesto vypísania celého refrénu napíšu len začiatočné slová a tri bodky za nimi. Nateraz sme sa to takisto rozhodli ignorovať a tri bodky sme z textov plne vymazali.

Čísla ako poradie slohy Tieto sme sa takisto rozhodli ignorovať a z pracovnej formy textu sme ich vymazali.

5 Základné porovnávanie

Aby sme vedeli podrobne vysvetliť náš algoritmus, musíme sa pozrieť na problematiku zvanú *Vector space model*. Idea je taká, že texty dokážeme reprezentovať tabuľkami rozmerov 2xn, kde prvá hodnota v každom stĺpci je slovo a druhá hodnota v prislúchajúcom stĺpci je početnosť tohto slova v dokumente. Tieto tabuľky dokumentov neskôr algebraicky porovnáme a zistíme ich takzvanú vzdialenosť v tomto modeli. Ku konkrétnostiam sa dostaneme v nasledujúcich podsekciách.

5.1 Poradie

Pozorný čitateľ mohol zistiť, že takýmto prístupom môžeme stratiť poradie slov. Preto sme implementovali vzor *TermScheme*, ktorý bude bližšie špecifikovať formu prvého riadku našich tabuliek. To znamená, že nebudeme v prvom riadku mať len slová, ale rafinovanejšie (usporiadané) skupiny slov, ktoré nám povedia aj kontextuálny význam slova. Využívali sme tieto schémy:

N-gram N-gram je všeobecne známy koncept, kde slová združujeme zaradom, ako sa v dokumente nachádzajú v skupinách o veľkosti N. V našom prípade sa budeme zaoberať un-, bi- a tri-gramami, teda skupinami dĺžky 1, 2 a 3. Vezmime napríklad vetu *"Ja som ukážková veta"*. Z tejto vety vieme vytvoriť:

1. Un-gramy
   (a) *Ja*
   (b) *som*
   (c) *ukážková*
   (d) *veta*
2. Bi-gramy
   (a) *Ja som*
   (b) *som ukážková*
   (c) *ukážková veta*
3. Tri-gramy
   (a) *Ja som ukážková*
   (b) *som ukážková veta*

Bez ujmy na všeobecnosti sa v nasledujúcich kapitolách budeme stotožňovať pojmy *skupina slov* a *slovo*.

5.2 Váženie slov

Prirodzený jav, ktorý v ľudskej reči nastáva je to, že rôzne slová sa v textoch vyskytujú častejšie a niektoré menej často. Toto vedie k úvahám, že slová majú pre nás rôznu dôležitosť. Existuje mnoho spôsobov, ako pojmom priradiť ich dôležitosť a teda ich numericky vyjadrenú váhu. My sme sa zaoberali týmito:

Tf Tf alebo *Term frequency* je zohľadnenie početnosti slova v danom dokumente. Najjednoduchšia implementácia tohto konceptu je slovu priradiť ako váhu jeho početnosť v dokumente. Iný prístup je početnosť ešte vyhladiť na základe dĺžky dokumentu aby sme zamedzili uprednostnením dlhších dokumentov, prípadne využiť logaritmus ako škálovací mechanizmus.

Idf Idf alebo *Inverse document frequency* je číselné vyjadrenie toho, ako často sa pojem nachádza vo všetkých dokumentoch a teda koľko informácie v sebe nesie. V slovenčine má napríklad slovo *a* váhu veľmi nízku, na druhej strane ľudové slovo *čarnica* má túto váhu omnoho vyššiu.

Tf-idf Tento systém berie do úvahy obe predošlé váhy.

5.3 Porovnávanie vektorov

Keď sme pred chvíľou spomínali tabuľky ako reprezentáciu dokumentov, tak sme vec trochu zjednodušili. Ide totižto o vektor váh slov. To, aby nám dimenzie sedeli zabezpečíme tým, že si tieto váhy zoradíme rovnako pre oba vektory na základe slov, ktorým prislúchajú. Tieto vyrovnané vektory potom porovnáme rôznymi algebraickými operáciami, ako je napríklad *kosínusová miera*.

6 Pokročilé porovnávanie

V tejto kapitole sa budeme venovať spôsobom, ako porovnávať slová s toleranciou a nie len podľa toho, či sa stringologicky rovnajú.

7 Tvorba grafického rozhrania

V tejto kapitole sa budeme venovať tvorbe grafického rozhrania.

8 Záver

Podarilo sa nám vymyslieť model a spôsob, akým náš algoritmus bude bežať. Takisto sme boli schopní nadizajnovať algoritmus tak, že bude bežať na viac fáz a každá fáza nám vráti iný (ale podobný) výsledok iným spôsobom. Pevne veríme, že touto prácou okrem iného pomôžeme udržaniu nášho pôvodného a autentického kultúrneho dedičstva.