# Dokumentácia projektu: PyLucene Music Search Engine

## 1. Popis projektu a motivácia

Hlavnou motiváciou projektu bolo vytvoriť vyhľadávač pre klaviristov, ktorý by im umožnil efektívne nájsť nové skladby na naučenie podľa rôznych kritérií ako skladateľ, žáner, obtiažnosť či časové obdobie. Cieľom bolo uľahčiť orientáciu v rozsiahlom repertoári klasickej klavírnej hudby.

Počas implementácie sa ukázalo, že pôvodná vízia pokročilého filtrovania podľa rýchlosti, podobnosti melodických vzorcov či dĺžky skladby nie je tak jednoducho realizovateľná, ako som si predstavoval. Tieto parametre nie sú štandardne dostupné v dátach a ich automatická extrakcia by vyžadovala komplexnú hudobnú analýzu, čo presahovalo rámec projektu.

Výsledný systém síce nedosahuje pôvodnú ambicióznu víziu, ale poskytol cenné poznanie o fungovaní vyhľadávacích systémov. Projekt mi umožnil prakticky pochopiť, ako funguje indexovanie dokumentov, váhování polí, textová analýza a vyhodnocovanie relevantnosti výsledkov. Naučil som sa pracovať s Apache Lucene, spracovávať dáta z Wikipédie pomocou Apache Spark a riešiť problémy s nekonzistentnými dátami.

Aj keď by som systém v jeho súčasnej podobe pravdepodobne sám aktívne nepoužíval na hľadanie repertoáru, projekt splnil svoj vzdelávací cieľ. Získal som praktické skúsenosti s problémami ako čistenie dát, spájanie záznamov s nekonzistentnými názvami skladieb a optimalizácia vyhľadávacích dotazov. Celková veľkosť datasetu je približne 5000 záznamov po spojení dát z PianoStreet.com s dátami z Wikipédie.

## 2. Prehľad existujúcich riešení

Pri analýze existujúcich riešení som identifikoval niekoľko platforiem pre vyhľadávanie klavírnej hudby:

**PianoStreet.com** - primárny zdroj dát pre môj projekt. Obsahuje katalóg skladieb s informáciami o skladateľovi, tonine, období a obtiažnosti. Dáta sú štruktúrované, ale často chýbajú podrobnejšie popisy a kontextové informácie.

**PianoLibrary.org** - dobre kategorizovaný zdroj, ale chýbajú mu textové popisy skladieb, čo limituje možnosti full-textového vyhľadávania. Vhodné skôr na browsing než na pokročilé vyhľadávanie.

**MuseScore.com** - primárne zameraný na zdieľanie notového materiálu. Obsahuje viacero verzií jednej skladby od rôznych používateľov, čo je výhodné pre notový materiál, ale komplikuje to jednoznačnú identifikáciu skladieb pre vyhľadávací systém.

**ClassicalArchives.com** - obsahuje rozsiahly katalóg kategorizovanej hudby, ale väčšinou ide o odkazy na externé zdroje.

**AllMusic.com** - veľmi kvalitný zdroj s podrobnými popismi a recenziami. Nevýhodou je, že obsahuje nielen klasickú hudbu a nie výhradne klavírnu tvorbu. S odstupom času si myslím, že keby som vedel vopred o obmedzenej veľkosti datasetu, použil by som túto platformu ako primárny zdroj.

**Musipedia.org** - teoreticky umožňuje vyhľadávanie podľa napísanej melodickej linky, čo by bolo inovatívne. V praxi sa mi však nepodarilo túto funkcionalitu spoľahlivo overiť.

Žiaden z existujúcich systémov neposkytuje komplexné riešenie kombinujúce full-textové vyhľadávanie, filtrovanie podľa obtiažnosti a časového obdobia s obohatenými informáciami z externých zdrojov ako Wikipedia.

# 3. Popis riešenia

## a) Architektúra riešenia

Systém je postavený na troch hlavných komponentoch:

1. **Spracovanie a obohacovanie dát** - extrahovanie informácií z Wikipédie pomocou Apache Spark (PySpark), čistenie a normalizácia dát, spájanie pôvodných dát s Wikipedia článkami na základe názvu a skladateľa.

2. **Indexovanie** - vytvorenie Lucene indexu s optimalizovanou štruktúrou polí, rozlišovanie medzi plnotextovo prehľadávanými poľami (TextField) a exaktne vyhľadávateľnými poľami (StringField), implementácia rozsahových dopytov pre roky a obtiažnosť (IntPoint).

3. **Vyhľadávanie** - multi-field search s váhovaním polí podľa dôležitosti, parsovanie a aplikácia filtrov (year ranges, difficulty levels), skórovanie a hodnotenie výsledkov.

## b) Použitý softvér

**Apache Spark (PySpark)** - použitý na distribuované spracovanie Wikipedia dump súboru:

- Filtrovanie miliónov článkov na relevantné skladby (compositions + piano kategórie)
- Parsovanie wiki pomocou regex operácií
- Token-based matching s Jaccard similarity
- Aggregate operácie pre deduplikáciu a validáciu
- Vytvorenie obohateného datasetu spojením troch zdrojov (matched, unmatched music, unmatched wiki)

**PyLucene** - Python wrapper pre Apache Lucene, jadro vyhľadávacieho systému:

- StandardAnalyzer pre tokenizáciu a normalizáciu textu
- Flexibilné indexovanie s podporou rôznych typov polí (TextField, StringField, IntPoint)
- MultiFieldQueryParser pre vyhľadávanie naprieč viacerými poľami
- Pokročilé query parsovanie s podporou boolean operátorov
- BooleanQuery pre kombináciu textových a range filtrov
- Relevance scoring s field boosting
- IntPoint range queries pre roky a obtiažnosť

**Docker** - kontajnerizácia celého riešenia:

- Zabezpečenie konzistentného runtime prostredia (Java, Python, PyLucene)
- Jednoduchá reprodukovateľnosť a nasadenie

**scikit-learn** - použitý pre vyhodnocovanie metrík (precision, recall, F1 score) na testovacích dopytoch.

## c) Problémy a ich riešenia

**Problem s implementáciou Spark** - Spark vyžaduje Java runtime a špecifickú konfiguráciu.

**Nekonzistentnosť názvov skladieb** - Najväčší problém projektu. Názvy skladieb v pôvodnom datasete a na Wikipédii sa líšia:

- `Etude in C Major, Op. 10 No. 1` vs `Étude Op. 10, No. 1 (Chopin)`
- `Fantasy in C Minor, K. 475` vs `Fantasia in C Minor, K. 475`
- `Für Elise in A Minor` vs `Für Elise`
- `Prelude & Fugue 1 in C Major, BWV 846 from The Well-Tempered Clavier Book 1` vs `Prelude and Fugue in C major, BWV 846`

Rozdíely zahŕňajú:

- Vymenené poradie slov
- Rôzny počet slov a detailov
- Odlišné zápisy (ampersand vs "and", skratky vs plné názvy)
- Prítomnosť/absencia mena skladateľa v názve
- Rôzne diakritické znamienka

**Nekonzistentnosť Wikipedia dát** - Infobox polia nie sú jednotné naprieč článkami. Niektoré skladby majú kompletné informácie, iné len čiastočné. Jediné konzistentné polia sú názov článku a prvý paragraf.

## d) Práce na projekte

**1. Spracovanie dát z Wikipédie (enrich_music.py):**

- Filtrovanie relevantných článkov o klavírnych skladbách (Category:Compositions + Category:Piano)
- **Hybridná tokenizácia:**
  - Compound tokeny pre katalógové čísla: `Op. 10 No. 2` → `op10no2`, `BWV 846` → `bwv846`, `K. 475` → `k475`
  - Compound tokeny pre toniny: `C Minor` → `cminor`, `F-sharp Major` → `fsharpmajor`
  - Normalizácia diakritiky (é→e, ü→u) a lowercasing
  - Rozdelenie zvyšného textu na jednotlivé slová (min. 2 znaky)
- **Token matching s validáciou:**
  - Jaccard similarity (threshold ≥ 0.3)
  - Minimálne 2 spoločné tokeny
  - Validácia skladateľa cez zátvorkový formát `(Composer)` v Wikipedia názve
  - Single-match filter (zamietnutie viacerých matchov 1 to many / many to 1)
- Extrakovanie štruktúrovaných informácií z Wikipedia infoboxov pomocou regex:
  - Composer, key, catalogue, opus, genre, form, composed year, movements
- Extrakovanie prvého odstavca článku pre textový kontext
- Čistenie (odkazy, šablóny, referencie, HTML tagy)
- Spojenie matched, unmatched music a unmatched wiki dát do jedného datasetu

**2. Vytvorenie indexera (indexer.py):**

**TextField + Field.Store.YES** - napr. pre názvy skladieb a skladateľov. Tieto polia sú plnotextovo prehľadávateľné a zároveň uložené pre zobrazenie vo výsledkoch. Vysoká váha pri vyhľadávaní.

**TextField + Field.Store.NO** - pre popisy, súhrny, Wikipedia paragrafy. Prehľadávateľné pre kontext, ale neukladané aby sa šetrilo miesto v indexe. Nie je potreba vidieť celý paragraf vo výsledkoch.

**StringField + Field.Store.YES** - pre toniny (key) a hudobné obdobia (period). Exaktné zhody bez tokenizácie. Napríklad "C Major" sa musí zhodovať presne, nie ako "c" a "major" samostatne.

**IntPoint** - pre roky a obtiažnosť. Umožňuje efektívne rozsahové dopyty typu "year:1800-1850" alebo "difficulty:hard" (level 6-10). Pridané aj NumericDocValuesField pre zoraďovanie.

**StoredField** - pre URL adresy. Uložené len pre zobrazenie, nie je potrebné ich prehľadávať.

**3. Vytvorenie vyhľadávača (searcher.py):**

**Multi-field search** - vyhľadávanie naprieč viacerými poľami súčasne (názov, skladateľ, Wikipedia text, žáner, forma).

**Váhovanie polí** - rôzne polia majú rôznu dôležitosť:

- name: 3.0 (najdôležitejšie)
- composer: 2.5
- wiki_title: 2.0
- wiki_paragraph: 1.5
- info_genre: 1.2
- description, summary: 1.0
- info_form, info_movements: 0.8

**Filtrovanie** - parsovanie špeciálnych filtrov z query stringu:

- `year:1850-1875` - rozsah rokov
- `difficulty:easy/intermediate/hard` - úroveň obtiažnosti (0-2 / 3-5 / 6-10)

**Vrátenie výsledkov** - triedené podle relevance score s možnosťou limitu počtu výsledkov.

**4. Evaluácia pomocou metrík (test_metrics.py):**

- Automatizované testovanie na sade queries.json
- Výpočet precision, recall a F1 score
- Identifikácia problematických dopytov

# 4. Popis dát

Dataset obsahuje nasledujúce polia:

**Pôvodné polia z PianoStreet.com:**

- `url` - odkaz na skladbu
- `composer` - skladateľ
- `name` - názov skladby
- `lead`, `group_name` - dodatočné kategorizačné polia
- `key` - tonina (napr. "C Major", "D Minor")
- `year` - rok vzniku
- `level` - obtiažnosť (1-10)

- `period` - hudobné obdobie (Baroque, Classical, Romantic, Modern)
- `description` - popis skladby
- `summary` - stručné zhrnutie

**Obohátené polia z Wikipédie:**

- `wiki_title` - názov Wikipedia článku
- `wiki_composer` - skladateľ podľa Wikipédie
- `wiki_paragraph` - prvý odstavec článku
- `info_composer` - skladateľ z infoboxu
- `info_key` - tonina z infoboxu
- `info_catalogue` - katalógové číslo (BWV, K., Op.)
- `info_opus` - opusové číslo
- `info_genre` - žáner
- `info_form` - hudobná forma (sonáta, etuda, prelude)
- `info_composed` - rok/obdobie kompozície
- `info_movements` - počet častí

**Charakteristika dát:**

- Niektoré záznamy obsahujú len pôvodné polia (bez Wikipedia dát)
- Niektoré obsahujú len Wikipedia polia (články, ktoré sa nepodarilo spojiť)
- Malá časť obsahuje oboje (úspešne spojené záznamy)
- Wikipedia dáta sú výrazne nekonzistentné - jediné konzistentné polia sú názov a paragraf
- Celkovo ~5000 záznamov

Dataset je dostupný ako `music_enriched.csv`.

# 5. Vyhodnotenie

## a) Konkrétne príklady

**Query: "mozart"**

- **Problem:** Slovo "mozart" sa môže vyskytovať v názve skladby (napr. "Variations on a Theme by Mozart"), čo má vyššiu váhu ako pole composer
- **Výsledok:** Nie všetky výsledky sú od Mozarta, niektoré len obsahujú jeho meno v názve
- **Riešenie:** Explicitné vyhľadávanie v poli composer alebo zmena váh

**Query: "bach minuet"**

- **Výsledok:** Funguje v poriadku
- **Dôvod:** Obe slová sú dostatočne špecifické a kombinujú composer + form

**Query: "romantic d-flat major difficulty:hard"**

- **Výsledok:** Väčšinou správne výsledky
- **Problem:** Tónina (key) nie je vždy konzistentná
- **Filtrovanie difficulty funguje správne**

**Query: "sonata year:1800-1810"**

- **Výsledok:** Funguje dobre
- **Rozsahové filtrovanie rokov pracuje správne**

**Query: "chopin waltz"**

- **Výsledok:** Väčšinou správne, ale jeden záznam nie je od Chopina - má len "chopin" v názve skladby
- **Ukazuje problém s váhovaním polí**

### b) Metriky

Testovanie bolo vykonané na súbore `queries.json` s definovanými očakávanými výsledkami. Script `test_metrics.py` automaticky vyhodnotil výkonnosť systému:

**Overall Metrics:**

- **Average Precision: 0.52** - V priemere polovica vrátených výsledkov je relevantná
- **Average Recall: 1.00** - Systém nachádza všetky relevantné dokumenty, ktoré existujú v datasete
- **Average F1: 0.62** - Harmonický priemer precision a recall

**Interpretácia:**

- Vysoký recall (1.00) znamená, že systém neprehliadne relevantné skladby
- Nižšia precision (0.52) znamená, že vo výsledkoch sú aj nerelevantné položky
- F1 score 0.62 je prijateľný pre experimentálny systém

**Možné zlepšenia:**

- Lepšie váhovanie polí na zníženie false positives
- Rozšírenie datasetu pre lepšie pokrytie
- Implementácia query expansion (synonymá, podobné termíny)

## 6. Inštalácia a použitie

### Spustenie

```
# Spustenie cez Docker Compose
docker-compose build
docker run --rm music-search
```

---

**Autor:** Marek Šebesta **Dátum:** December 2025
**Repozitár:** https://github.com/sebestam/Projekt-VINF