Information Retrieval: Booleovský model

Soukup Jan

Popis projektu

Můj projekt se zabývá získáváním informací pomocí booleovského modelu (včetně preprocessingu dat, indexace).

Vstupem aplikace je nějaký booleovský dotaz, který je zadán ve webovém interface. Vstup se skládá z termů a logických spojek, spojky musí být zapsány slovně, tedy AND, OR, NOT (na velikosti písma nezáleží). Zároveň může být výraz jakkoliv uzávorkován.

Výstupem aplikace je seznam dokumentů(písniček) nacházejících se v databázi. Výstup není limitován, zobrazí se všechny nalezené písničky splňující dotaz. Uživatel může dále rozkliknout jednotlivé písničky a získat informace k textu.

Způsob řešení

Booleovský model je jeden ze způsobů, jak prohledávat kolekci dokumentů s cílem identifikace dokumentů obsahujících (či neobsahujících) termy, které odpovídají dotazu. Dotaz je tvořen booleovským výrazem. Booleovský model vyhledává dokumenty s přesnou shodou. Dotaz je vyhodnocován vůči kolekci dokumentů – každý dokument můžeme tedy chápat jako objekt v databázi. Každý dokument nejdříve projde fází preprocessingu – odstraněna nevýznamná slova (slova, která nesou málo informací, např. spojky, předložky). Významná slova jsou dále stemmována nebo lemmatizována, abychom získali základní tvary slov. V booleovském modelu ukládáme každý dokument jako binární vektor (označuje, zda v něm slovo je, či není), tím dostáváme term-by-document matici, kde na i-tém řádku v j-tém sloupci je 1 právě tehdy, když je term obsažen v dokumentu. Tato informace nám ale neříká, kolikrát je daný term obsažen v dokumentu. Samotné uložení do matice je velmi paměťově neefektivní – dokumenty můžou být naprosto odlišné, tím získáme velké množství termů a v matici budeme mít 1 jen zřídka. Proto ukládáme informace, v jakých dokumentech se nachází term v invertovaném seznamu. Invertovaný seznam obsahuje seznam termů a ke každéu termu je přiřazen seznam dokumentů, ve kterém se term nachází. Díky invertovanému seznamu pak můžeme pro booleovské operace využít algoritmy z přednášky, kde procházíme dva listy a vůči operaci rozhodujeme, zda přidáme dokument do výsledného seznamu, nebo ne.

Implementace

Aplikace se skládá z frontendu, backendu a databáze.

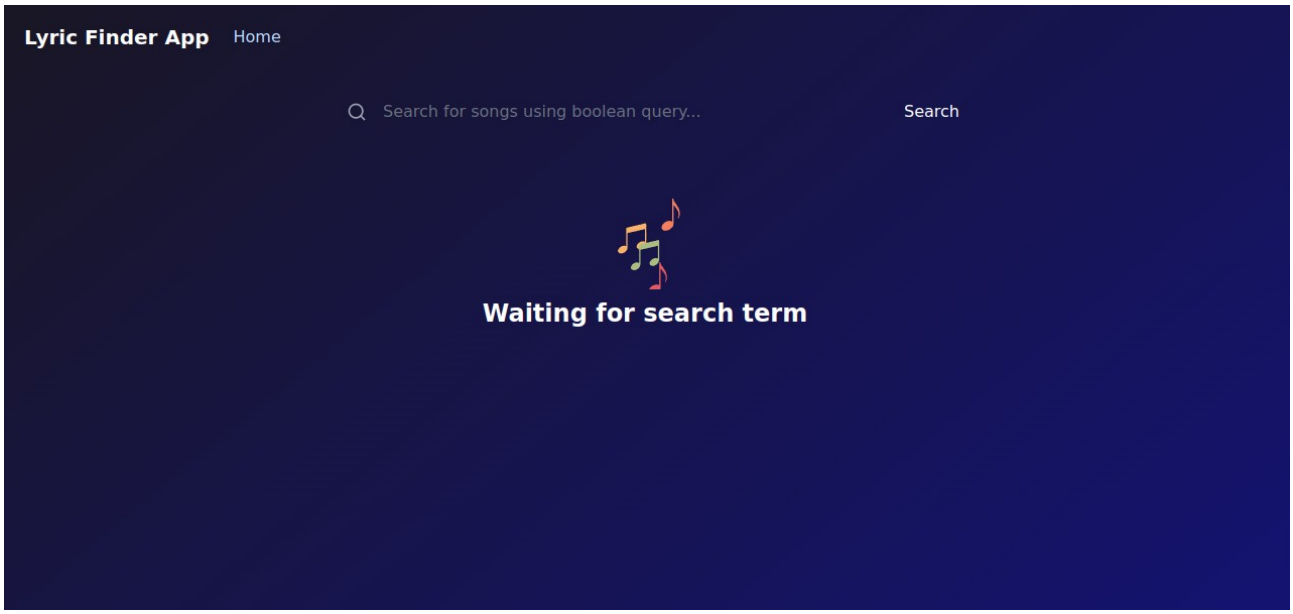
Frontend je psaný ve frameworku React v jazyce Javascript. Tato část aplikace slouží jen jako rozhraní pro uživatele. Hlavním prvkem je searchbar, do kterého se zadává booleovský výraz. Výraz musí být zadaný ve slovním formátu, tedy booleovské spojky musí být ve tvaru AND, OR, NOT. Po odeslání výrazu se zobrazí nalezené písničky, jejichž text splňuje daný výraz. Každá z nalezených písniček lze dále rozkliknout dále do detailu, kde si může uživatel přešíst celý text.

Backend je psaný ve frameworku FastApi v jazyce Python. Backend pracuje jako API. Endpoint /songs/id vrací data ke konkrétní písničce. Endpoint /songs zprostředkovává parsování dotazu, vyhodnocení dotazu s pomocí databáze a invertovaného seznamu a následného odeslání nalezených písniček. Během zapnutí backendu také dochází k vytvoření invertovaného seznamu, stemmovaného seznamu a import písniček společně s invertovaným seznamem do databáze. Pro extrakci pojmů, stemming a lematizaci jsem využil knihovny nltk pro python, která velmi ulehčila

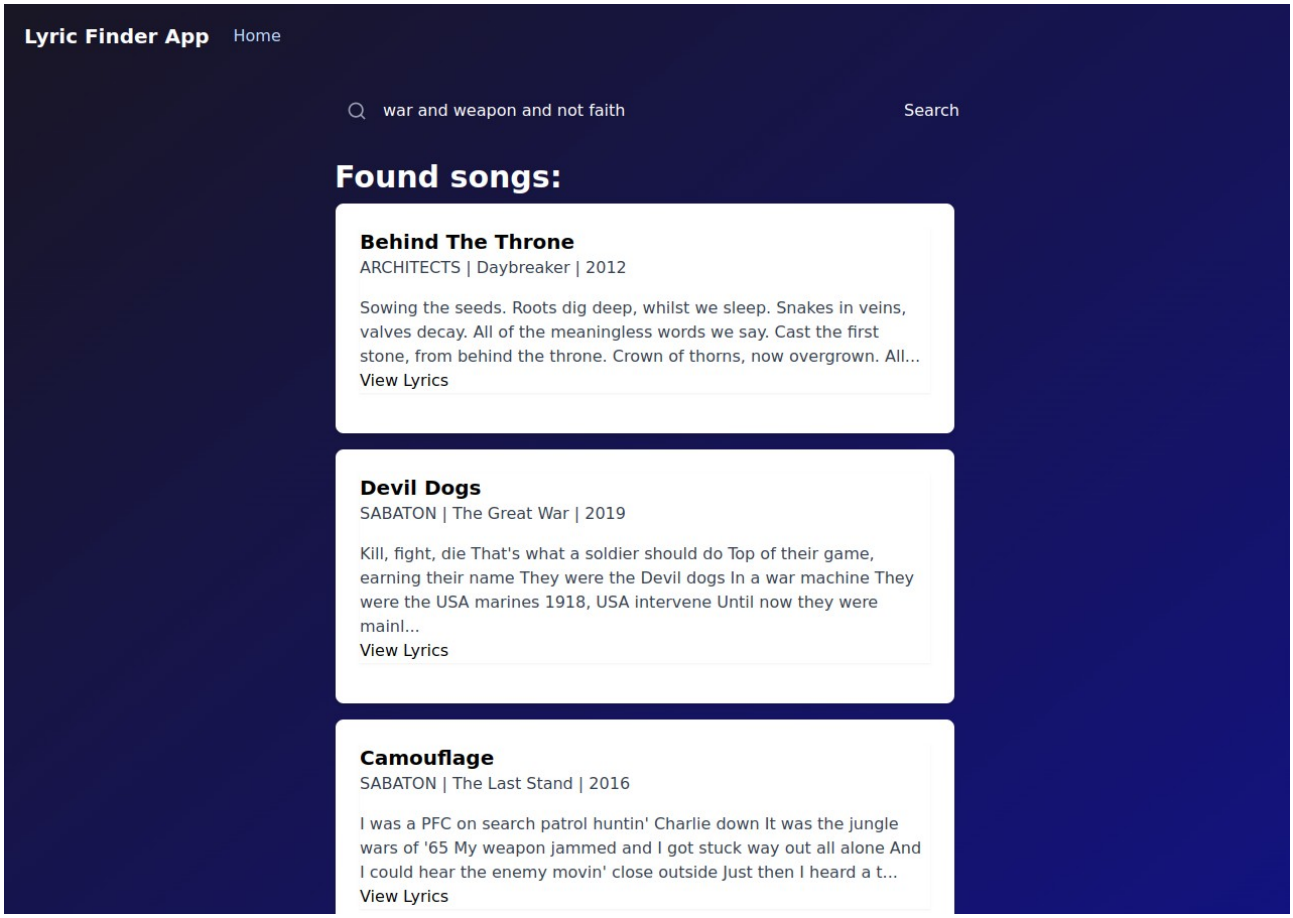
práci s identifikací díky seznamu stopwords pro identifikaci nevýznamných slov a také jsem použil PorterStemmer, který zajistil stemming významných slov. Tyto data jsou uloženy do .json souborů a následně vloženy do databáze. Invertovaný seznam je sestaven jako výčet termů a seznam písniček, v jakých se term vyskytuje. Parsování booleovského dotazu je provedeno tak, že se dotaz nejdříve převede do prefix formátu pro jednodušší práci s ním a následně se prochází podle aktuální booleovského operátoru. Po vyhodnocení booleovského výrazu získáme list dokumentů, které splňují dotaz, následně je získáme z databáze a odešleme.

Databáze je lokální typu PostgreSQL. K práci s databází se využívá sqlalchemy pro python. V databázi jsou uloženy písničky a invertovaný seznam, které do ní jsou při spuštění aplikace načteny.

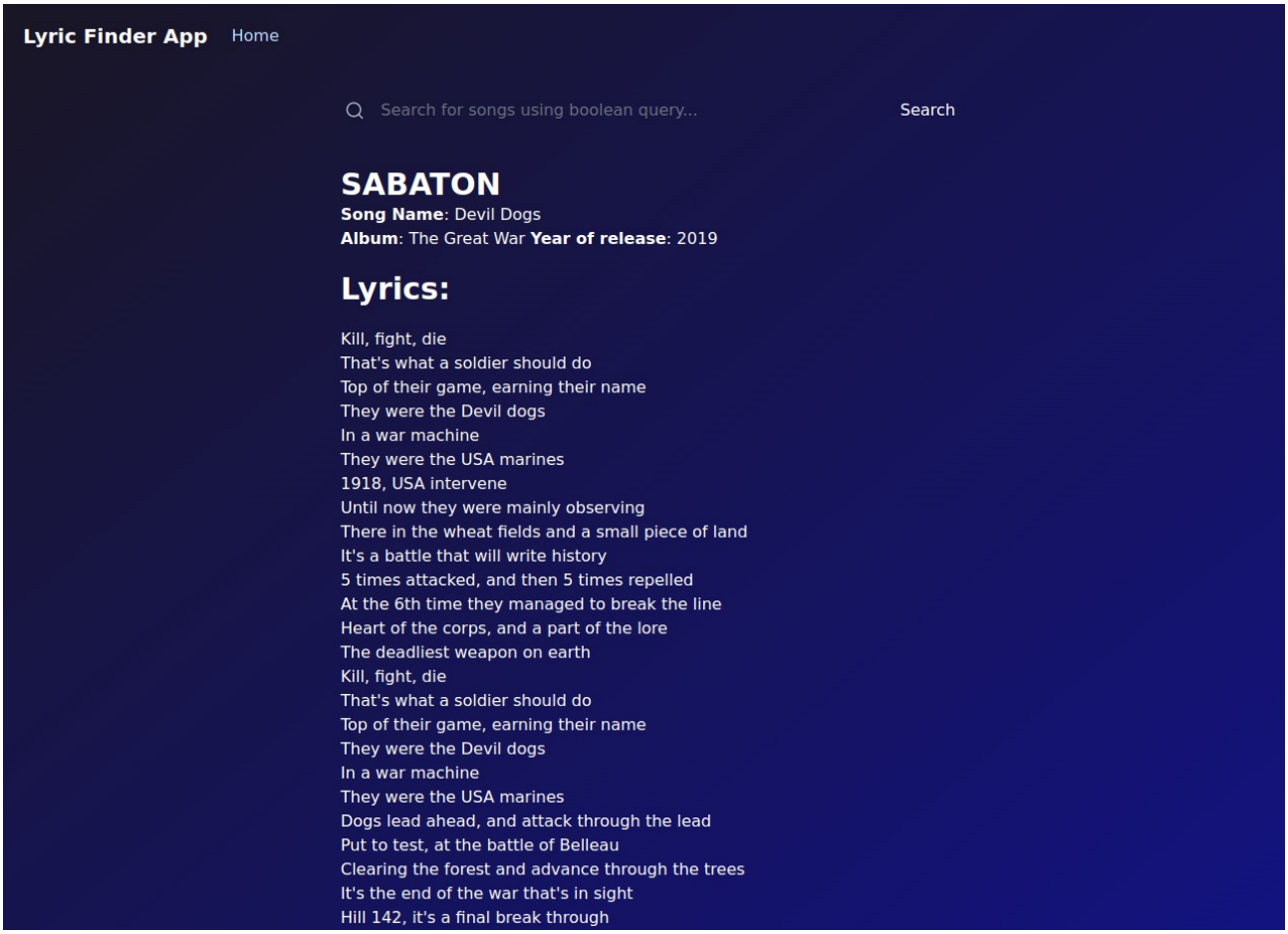
Příklad výstupu



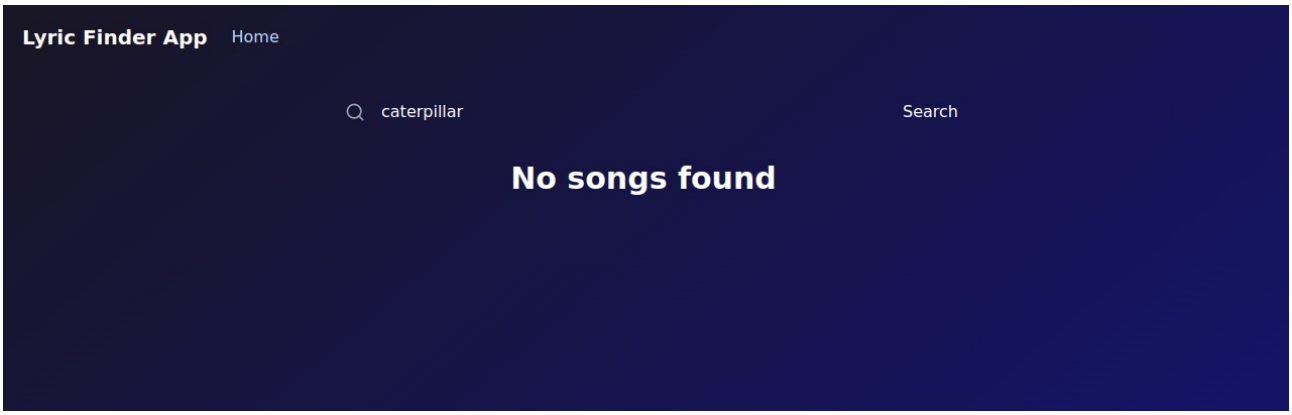
Hlavní stránka po spuštení aplikace



Nalezené písničky pro daný dotaz



Detail písničky



Nenalezení žádné písničky vůči dotazu

Experimentální sekce

V této sekci se pokusím zaměřit na rychlost vyhledávání v booleovském modelu za využití invertovaného seznamu (v databázi) oproti naivnímu řešení, které sekvenčně prochází všechny písničky v databázi a zjišťuje, zda je hledaný term v textu. K tomuto účelu jsem zkusil vymyslet několik možných dotazů na můj vybraný soubor písniček. Čas je měřený na backendu v pythonu za pomocí utility time, která měří dobu běhu funkce pro nalezení všech dokumentů pro obdržený výraz. Funkce zahrnuje parsování výrazu, převedení výrazu do prefixové formy a následné vyhledávání dokumentů pomoci invertovaného listu, nebo za pomoci sekvenčního vyhledávání v databázi písniček za pomocí query v postgresql.

Tabulka

Dotaz	Inverted List [ms]	Sequential [ms]
war	9.46	10.38
(war and not dog)	10.96	25.72
(war and not dog) or time	15.42	31.48
(war and not dog) or (time and not fight)	18.86	35.14
(war and not dog) or (time and not fight) and boom	22.62	36.62
(war and not dog) or (time and not fight) and boom or wisdom	23.83	52.68
(war and not dog) or (time and not fight) and boom or (wisdom and not world)	29.23	62.40
not ((war and not dog) or (time and not fight) and boom or (wisdom and not world))	33.89	78.94

Diskuze

Moje práce není dokonalým řešením zadaného problému. Při testování jsem došel k závěru, že jsem měl moc malý dataset, na kterém se pak rychlejší booleovský model moc neprojeví. Bylo by také možné podporovat více vstupů pro dotazy, např. místo jen slovních operací povolit i znaková, také by bylo lepší více kontrolovat vstupy. To by bylo potřeba řešit buď na frontendu či při parsování samotného dotazu.

Závěr

Implementace byla o dost náročnější, než jsem původně očekával. Díky dostupným knihovnám jsem se rozhodl na backend zvolit Python v kombinaci s FastApi. Tato práce byla moje první seznámení s Pythonem a byl to dost velký boj. Samotná práce s vyhledáváním v dokumentech byla velmi zajímavá.