

Dokumentace pro semestrální práci  
“Booleovský model“ v rámci předmětu  
“Vyhledávání na webu a v multimediálních  
databázích“ (BI-VWM)

Dmytro Osaulenko

23. června 2020

## Popis projektu

Cílem projektu bylo vytvořit implementaci teoretického (booleovského) modelu domény získávání informací (“information retrieval”), jehož hlavní myšlenkou je vyhledávání specifických slov (termů) ve statické kolekci textů (většinou předem zpracovaných pro usnadnění a zrychlení vyhledávání) pomocí spojek výrokové logiky jako jsou *AND*, *OR*, *NOT*. Uživatel by na takový vstup dostal jako výstup množinu dokumentů, pro které uvedený logický výraz platí (případně by nedostal, pokud nějaký z termů by v kolekci nebyl přítomen, nebo by výraz z hlediska výrokové logiky nebyl platný).

## Způsob řešení

První fází řešení je nalezení vhodné (podle velikosti a obsahu) kolekce textů (případně kolekcí textů). Následuje předzpracování (“preprocessing”) pro optimalizaci vyhledávání a ukládání zpracovaných dat. Jedná se hlavně o techniky NLP (“Natural language preprocessing” nebo česky “Zpracování přirozeného jazyka”) jako tokenizace (“tokenization”), tj. extrakce jednotlivých slov (termů) z textu (bez interpunkce atd.), a stematizace (“stemming”), tj. nalezení kmene slova, nebo sofistikovanější (ale i dražší/náročnější) operace - lematizace (“lemmatization”), tj. vyhledání základního tvaru slova.

Z mnoha variant ukládání a pak i následujícího dotazování mnou bylo zvoleno ukládat data do *hašovacích tabulek* (jinými slovy abstraktního datového typu (ADT) *hašovací mapa* (*hash map*) neboli *slovník* (*dictionary*)) anebo ještě přesněji - NoSQL databáze, co poskytuje takovou možnost. Tak by pro každou statickou kolekci textů existovaly 2 mapy (pro optimalizaci místa): jedna mapa by reprezentovala dokumenty v kolekci (*klíč* - unikátní identifikátor dokumentu, *hodnota* - název dokumentu), druhá by reprezentovala prezenci termů v dokumentech (*klíč* - jednotlivý (stematizovaný) term, *hodnota* - množina identifikátorů dokumentů pro které platí, že term je přítomen v dokumentu).

Po zadání uživatelem dotazu by se ten vyhodnotil knihovnou třetí strany pro výrokovou logiku a byl převeden na problém typu SAT (“satisfiability problem”, česky “problém splnitelnosti”). SAT solver knihovny by dotaz zpracoval a nabídl všechny možné varianty (pokud jsou) jak výraz splnit a podle toho by se logická část implementace dotazovala databáze s hash mapami, sestavila by výsledek a vrátila ho uživateli.

## Implementace

Z různých možností a variant pro uživatelské rozhraní jsem se rozhodl pro jednoduchou webovou stránku (z mého pohledu nejpohodlnější a nejprívětivější pro uživatele možnost). Uživatelovi stačí vědět (a zadat v prohlížeči) webovou adresu aplikace a hned ji může používat na jakémkoli svém chytrém zařízení.

Jako architekturu aplikace jsem zvolil klient-server. Frontendová část (klient) je implementovaná pomocí populárního Javascript-frameworku *Vue.js* a frontend knihovny CSS-stylů Bootstrap 4. Klient řeší takové úkoly, jak je příjem dotazu

od uživatele (specifikovaná kolekce textů a booleovský výraz termů), komunikace se serverem (odesílání uživatelského požadavku a příjem odpovědi serveru), zobrazení výsledků uživateli. Knihovna *Bootstrap 4* řeší mimo jiné *responsive design*, takže aplikaci se dá používat i z chytrého telefonu.

Backendová část (server) je implementována pomocí jazyka *Python* a využívá takové knihovny třetích stran jako:

- NLTK - implementace stematizace PorterStemmer;
- PyEDA - parsování a validace booleovského výrazu, SAT solver;
- aiobotocore - asynchronní verze knihovny *botocore* pro komunikaci se službami Amazon Web Services (AWS) (NoSQL databáze - AWS DynamoDB, úložiště souborů - AWS S3, výpočtová serverless jednotka - AWS Lambda).

Zpočátku bylo potřeba najít texty a zpracovat je. Rozhodl jsem se, že jednodušší variantou jak pro mě, tak i pro uživatele budou texty v anglickém jazyce. Potom jsem chtěl nemít pouze jedinou kolekci textů, ale alespoň několik a to z různých oborů a různé velikosti, a to jak pro zajímavost dotazu/výsledků, tak i pro účely vývoje a ladění. Moc velká kolekce textů by se jednak dlouho zpracovávala a potom by se v ní hledalo pomaleji, než v menší, a u menší kolekce by nastal problém řídkých a nezajímavých výsledků.

Zvolil jsem 3 kolekce různé velikosti a stylu: menší dump z anglické Wikipedie (největší kolekce, cca 15 tisíc článků, cca 250 MB), 100 knih klasické světové literatury (beletrie), taky anglicky, přesněji tedy 100 knih ze stránky “Top 100 EBooks yesterday“ projektu Gutenberg (cca 60 MB) a nejmenší kolekci je bibliografie moderního amerického spisovatele Chucka Palahniuka (cca 20 knih, cca 6 MB). Všechny soubory byly ve formátu .txt, nic méně skript, co zpracovává texty používá knihovnu *textract* co umí raw text vytáhnout ze souborů různých formátů (např. .pdf, .docx, .epub atd.), takže tato implementace tuto možnost také podporuje.

Během vývoje a ladění jsem používal lokální (vývojářský) server na backendu (knihovna *aiohhttp*), frontend byl hostovan také lokálním serverem pro frontend. První experimenty se prováděly nad soubory formátu JSON a potom i v NoSQL databázi běžící na lokálním počítači (např. MongoDB). Nic méně pro účel zveřejnění a fungování webové aplikace bylo potřeba ji nasadit (“to deploy“) do produkčního prostředí. Rozhodl jsem se používat služby Amazon Web Services (dále AWS), a to AWS Lambda pro backend, AWS DynamoDB jako NoSQL databázi, AWS S3 pro hosting statického webu. Navíc jsem implementoval možnost po obdržení výsledků jednotlivé dokumenty si prohlédnout, případně stáhnout, dokumenty příslušné kolekce jsou také uloženy na AWS S3. Aby frontend (z S3 bucketu) mohl komunikovat s “lambdou“ bylo nastaveno REST API pomocí AWS ApiGateway. Celkové schéma aplikace bude znázorněno trochu níže na obrázku.

Bohužel ve zdrojových kódech (kvůli časovým možnostem) není verze pro úplně lokální běh aplikace, bez využití jakýchkoli “cloud“ služeb (až na lokální server místo AWS Lambda), hodně se to spoléhá na DynamoDB a S3.

<input checked="" type="radio"/> WikipediaDump_files	
<input type="radio"/> WikipediaDump_terms	

<input type="checkbox"/>	154	Emperor Kazan.txt
<input type="checkbox"/>	187	Armed forces of the Netherlands.txt
<input type="checkbox"/>	251	Polymorphism.txt
<input type="checkbox"/>	1007	Alexis Korner.txt
<input type="checkbox"/>	1175	Educational essentialism.txt
<input type="checkbox"/>	1259	Marbury v Madison.txt
<input type="checkbox"/>	1581	Pope Gregory XV.txt
<input type="checkbox"/>	1838	List of NATO reporting names for submarines.txt
<input type="checkbox"/>	2289	Borneo.txt
<input type="checkbox"/>	2485	Luigi Pirandello.txt
<input type="checkbox"/>	2529	Posthumanism.txt
<input type="checkbox"/>	2544	Lake Nicaragua.txt

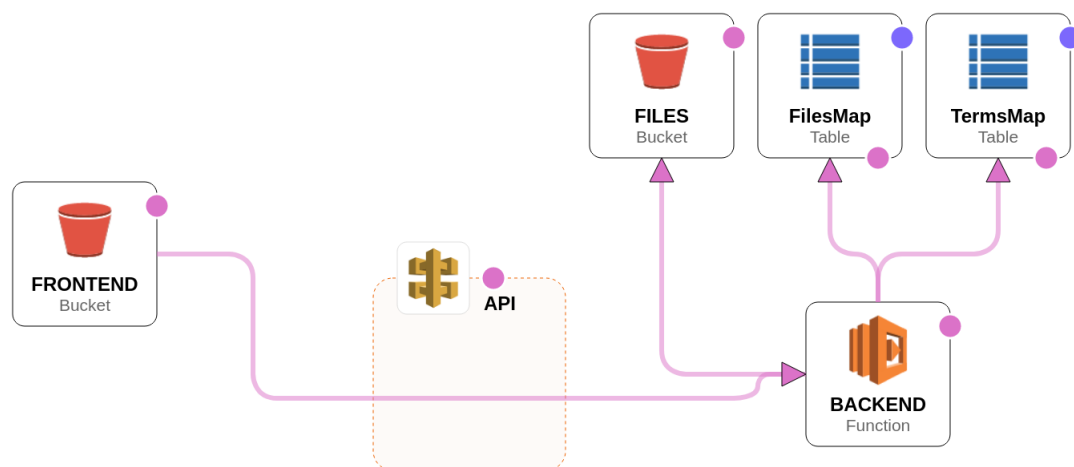
Obrázek 1: DynamoDB, kolekce WikipediaDump, mapa identifikátor dokumentu (klíč): název dokumentů (hodnota)

<input type="radio"/> GutenbergeTOP100_files	
<input checked="" type="radio"/> GutenbergeTOP100_terms	

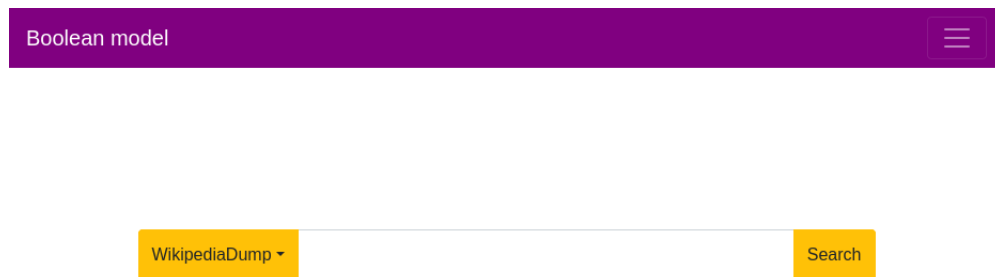
term	files
faintli	{ 0, 64, 31, 62, 29, 27, 58, 23, 54, 21, 17, 48, 15, 13, 42, 9, 70, 7, 38, 69, 3...
final	{ 0, 62, 29, 25, 54, 21, 17, 46, 13, 9, 71, 38, 50, 5, 67, 10, 34, 1, 63, 59, 26,...
frisk	{ 0, 53, 18, 13, 45, 42, 60, 50, 30, 3, 67, 33, 65 }
function	{ 0, 31, 61, 28, 59, 25, 24, 23, 22, 54, 53, 15, 47, 45, 70, 39, 37, 69, 3, 67, ... }
furi	{ 0, 62, 29, 25, 54, 21, 17, 9, 38, 50, 5, 10, 34, 1, 63, 59, 26, 55, 22, 18, 14,...
fusili	{ 0, 17, 25 }
habsburg	{ 0 }
hauck	{ 0 }
hipshak	{ 0 }
holi	{ 0, 31, 27, 23, 54, 21, 19, 17, 48, 15, 44, 11, 9, 70, 7, 71, 50, 69, 30, 36, 3,...
imperson	{ 0, 62, 28, 27, 24, 22, 53, 15, 9, 8, 38, 50, 69, 30, 20, 35, 67, 65 }
implicit	{ 0, 31, 61, 28, 59, 24, 48, 47, 13, 9, 70, 7, 38, 69, 34, 33 }

Obrázek 2: DynamoDB, kolekce GutenbergTOP100, mapa term (klíč): množina identifikátorů dokumentů (hodnota)



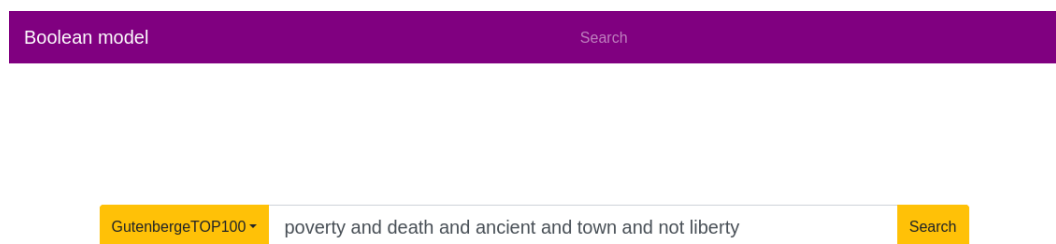
Obrázek 3: Schéma architektury finální nasazené aplikace: frontend (AWS S3 bucket) volá backend AWS Lambda přes AWS ApiGateway. Lambda-backend dělá dotaz na tabulky v DynamoDB a vrací spočítaný výsledek na frontend. Pokud uživatel požádá o zpřístupnění jednotlivého dokumentu, frontend odešle takový požadavek backendu a backend vrátí link (expirující za 5 minut) na požadovaný soubor.

## Příklad výstupu



The screenshot shows a search interface with a purple header bar containing the text "Boolean model" and a hamburger menu icon. Below the header, there is a search bar with a yellow dropdown menu on the left showing "WikipediaDump" and a yellow "Search" button on the right.

Obrázek 4: Hlavní stránka vyhledávání. Do příslušného pole uživatel zadává hledaný výraz. V levém dropdownu lze zvolit kolekci textů, nad jakou bude dotaz proveden.



The screenshot shows the same search interface as before, but with the dropdown menu now showing "GutenbergeTOP100". The search bar contains the text "poverty and death and ancient and town and not liberty" and the "Search" button is still present.

Obrázek 5: Zadání výrazu. Po kliknutí na *Search* se dotaz odešle logické části, ta jeho zhodnotí a vrátí výsledky.

GutenbergTOP100 ▾
poverty and death and ancient and town and not liberty
Search

Anne of Green Gables by L. M. Montgomery (14398).txt
Download

A Christmas Carol in Prose; Being a Ghost Story of Christmas by Charles Dickens (4128).txt
Download

Songs of Innocence, and Songs of Experience by William Blake (6399).txt
Download

The Scarlet Letter by Nathaniel Hawthorne (14639).txt
Download

Parsed (possibly simplified) query (DNF): And(poverty, death, ancient, town, ~liberty)

Time: 0.002965688705444336 s

Obrázek 6: Výsledky, pro které dotaz (logický výraz) platí. Při kliknutí na *Download* může si uživatel dokument prohlédnout.

## Experimentální sekce

U malých dotazů (např. pouze “dog“) čas po stisknutí *Search* je o hodně větší, než u dotazů s více termíny. Pokud nějaký termín nebude v mapě existovat, ale bude například součástí výrazu s velkým množstvím AND spojek, vyhodnocení výsledků bude o hodně rychlejší. Nic méně u malých kolekcí textů ten rozdíl zas tak velký ale nebude.

Příklady:

Dotaz	Kolekce	Čas [s]
dog	WikipediaDump	19.88
dog and cat	WikipediaDump	4.76
foo or foobar	WikipediaDump	0.98
not (dog or cat)	WikipediaDump	0.34
programming and computes and python and javascript	WikipediaDump	1.00

Tabulka 1: Dotazy, kolekce textů, čas na vyhodnocení dotazu

Výše uvedené časy jsou použity z footeru stránky po vyhodnocení a obdržení výsledků. Časem vyhodnocení je míněn rozdíl času od zahájení vyhodnocení výrazu po vrácení výsledků.

## Diskuze

Hlavním nedostatkem booleovského modelu je absence ceny termínů. Tento nedostatek je vyřešen v tzv. *Rozšířeném booleovském modelu* (“Extended boolean model“).

Jako rozšíření zadání jsem chtěl udělat možnost mít dynamickou kolekci textů

(s přidáváním nových dokumentů nebo mazáním starých) nebo i vytvářet nové kolekce, nic méně tato rozšíření by potřebovala další čas pro vývoj.

## Závěr

Jsem rád, že jsem se v předmětu *BI-VWM* blíž seznámil s doménou *information retrieval*, s jedním z její teoretických modelů - *booleovským* a měl jsem zkušenost ho implementovat.

Bavilo mě to a ohromnou část času, co jsem u toho ztrávil, jsem spíše vymýšlel rozmanitá rozšíření zadání a implementace, než samotný algoritmus.

Nakonec jsem se k zadání vrátil a to dodělal, i když ne tak, jak jsem to začátku viděl a chtěl, ale jsem s finální implementací spokojen.

Nic méně díky této úloze mám inspiraci dělat ve volném čase projekty podobného typu.