# Rozšířený boolovký model

*Vypracovali: Tomáš Peterka a Jakub Richtár*

## Popis projektu

Projekt je implementací rozšířeného boolovského modelu. Tento projekt tedy uživateli umožňuje na základě zadaných termů vyhledat co možná nejrelavantnější dokumenty.

Uživatel ve webovém prostředí zadá boolovský dotaz, který se bude skládat z dílčích termů, závorek a logických operátorů (AND, OR, NOT). Aplikace term zpracuje a zobrazí názvy dokumentů seřezené sestupně dle váhy (tedy relevance dokumentu vůči zadanému termu). Navíc uživateli bude umožněno si obsah konkrétních dokumentů v prohlížeči zobrazit.

## Způsob řešení

### 1) Získání dokumentů a jejich zpracování

Kolekci dokumentů jsme získali ze stránky https://anc.org/data/oanc/download . V kolekci se nacházelo něco kolem 62 000 dokumentů, kde každý dokument mel několik stovek až tisíc slov. Nechali jsme tedy jen 5000 vyhovujících dokumentů (200 až 1000 slov). Z těchto dokumentů jsme následně provedli extrakci termů, tedy vytvořili jsme kopii kolekce a pro každé slovo vymezili jeden řádek. Pak už jen následovalo odstranění nevýznamových slov (předložky, spojky, členy atd.) a tzv. „lematizace“ (úpravu na základní tvar slova).

### 2) Invertovaný seznam

Po úpravě dokumentů jsme tedy získali kolekci termů. Abychom ale mohli realizovat vyhledávání, museli jsme nějakým způsobem tyto termy uložit. Nabízela se „term-by-document“ matice, kde řádek reprezentuje term, sloupec název (id) dokumentu a hodnotou je pak vypočítaná váha (míra relevance) termu vůči dokumentu. Tato reprezentace by avšak vedla na 2D pole ve kterém bychom mohli vyhledávat pouze v O(n^2).

Zvolili jsme tedy jinou reprezentaci, a to invertovaný seznam. Ten je založený na principu takové struktury, která obsahuje všechny termy jako klíče a přiřazenou hodnotou je pak seznam dokumentů, ve kterých se daný term nachází a u každého dokumentu je ještě navíc uvedená váha termu. Tato reprezentace dále vedla na mapu (dictionary), ve které jsme schopni vyhledávat v O((log(n))^2).

### 3) Výpočet vah

V textu jsme již několikrát zmínili váhu v kontextu míry relevance termu vůči dokumentu. Výpočet takové váhy stojí na tzv. „tf-idf“ schématu, které je založeno na frekvenci výskytu termu v dokumentu a výskytu termu ve všech dokumentech.

Uvažujeme-li váhu termu i v dokumentu j pak vzorec pro výpočet váhy $w_{ij}$ je:

**$w_{ij} = tf_{ij} * idf_i = f_{ij} / max(f_{ij}) * log_2(n/df_i)$, kde**

n = počet všech dokumentů v kolekci

$f_{ij}$ = počet výskytů termu i v dokumentu j

$df_i$ = počet dokumentů obsahujících term i

$max(f_{ij})$ = maximální počet výskytu termu přes všechny dokumenty

## 4) Způsob vyhledávání

Pro tento projekt se nabízely 2 možné způsoby pro vyhledávání dokumentů. Oba způsoby využívají invertovaného seznamu popsaného výše, ale každý ze způsobů s ním pracuje jinak.

Sekvenční způsob prochází celým seznamem, dokud nenarazí na zadaný term a poté projde celým seznamem souborů v daném termu, dokud nenarazí na daný soubor. Jako výsledek vrátí váhu zadaného termu pro daný soubor. Časová složitost tohoto vyhledávání je O(n^2).

Druhý způsob je mnohem rychlejší, jelikož využívá přímo principu struktury invertovaného seznamu. Nejprve se v seznamu vyhledá zadaný term jako klíč, dále se pak v seznamu dokumentů tohoto termu vyhledá soubor jako klíč. Časová složitost tohoto vyhledávání je mnohem nižší než u sekvenčního způsobu, konkrétně O((log(n))^2).

V našem projektu jsme implementovali oba způsoby vyhledávání a umožnili tak výběr způsobu na uživateli.

## 5) Vyhodnocení výsledné váhy zadaného výrazu

Výslednou váhu vypočítáme na základě operátorů, který zadaný výraz používá. Pro jednotlivé operátory jsme použili následující vzorce:

- Pro dotaz typu $q = m_1 \,\&\, m_2$ (operand AND)

$$\mathrm{relev}(q, d_j) = 1 - \sqrt{\frac{(1-w_{1,j})^2 + (1-w_{2,j})^2}{2}}$$

- Pro dotaz typu $q = m_1 \mid m_2$ (operand OR)

$$\mathrm{relev}(q, d_j) = \sqrt{\frac{(w_{1,j})^2 + (w_{2,j})^2}{2}}$$

- Pro dotaz typu $q = !\, m_1$ (operand NOT)

$$\mathrm{relev}(q, d_j) = 1 - w_j$$

# Implementace

## 1) Preprocessing dokumentů

Abychom zajistili správné fungování odstranění nevýznamových slov a lemmatizace, tak jsme nejdříve všechna písmena v dokumentech převedly na lower case (makeLowerCase.py). Následovalo využití NLTK knihovny pro odstranění stop words, konkrétně jsme využili zdroje 'punkt' a 'stopwords' (removeStopWords.py).

Posledním krokem preprocessingu byla tokenizace a využití NLTK zdroje 'wordnet' pro lemmatizaci (stemming.py). Výsledkem tohoto procesu byla kolekce termů, kde každý řádek v daném dokumentu obsahoval jeden term.

## 2) Vytvoření invertovaného seznamu

Při vytváření invertovaného seznamu jsme zhotovili hned několik úkonů najednou (inverted_index.py). Nejdříve jsme zvolili python datovou strukturu dictionary pro samotnou reprezentaci. Jako klíč jsme zvolili string v podobě názvu termu a jako value jsme zvolili další dictionary, kde klíč je opět string, ale tentokrát reprezentuje název dokumentu, value tohoto dictionary je pak váha termu v tomto dokumentu. Po zvolení této struktury jsme vypočítali frekvenci dokumentů a frekvenci příslušných termů.

Následoval výpočet vah termů dle specifikovaných vzorců výše. Při výpočtu jsme výsledky rovnou zapisovali do našeho invertovaného seznamu, jehož strukturu jsme již popsali.

Nakonec jsme využili knihovny sklearn.preprocessing konkrétně MinMaxScaler pro normalizaci vah. Získali jsme tedy pole znormalizovaných vah, které jsme proiterovali a dříve vypočítané váhy přepsali.

Jako konečný datový typ invertovaného seznamu jsme zvolili JSON, kvůli jednoduchému nahrání seznamu do ostatních python programů.

## 3) Kontrola vstupu

Při získaní vstupu nejdřív kontrolujeme, zda je zadaný vstup správný. Za špatný vstup považujeme:

- Prázdný výraz a/nebo nevybraný způsob vyhledávání
- Obecně neplatný booleovský výraz (špatné umístění operátorů a termů apod.)
- Používání ve výrazu jiné než alfanumerické znaky (kromě použití závorek)
- Špatné uzávorkování výrazu

V takovém případě se bez jakéhokoliv vyhledávání zobrazí chybová hláška.

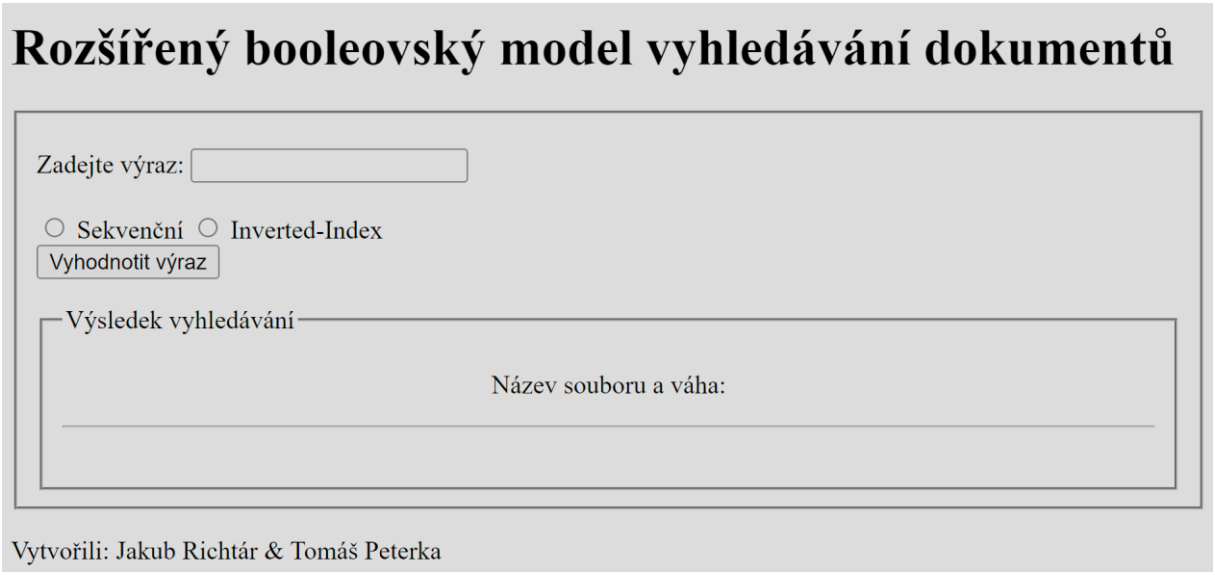
Pokud je zadaný výraz platný, pošleme ho dále na další zpracování.

## 4) Parsing a query

Před samotným výpočtem vah výrazu jsme si nejdříve museli výraz zpracovat, abychom mohli korektně váhu vypočítat dle priorit operátorů a uzávorkovaných podvýrazů. Pro takové zpracování výrazů jsme si naimplementovali vlastní funkci, která vrací seznam, který obsahuje nejprve všechny operátory seřazené podle priority, následované seřazenými termy. Tuto tabulku si převezme rekurzivní funkce, která postupně načte celý seznam, nejprve tedy operátory, díky kterým vytvoří výsledný vzorec (viz. výše) a poté do vzorce dosadí jednotlivé termy. Výsledkem celého procesu je váha daného souboru pro zadaný výraz. Pokud je váha nenulová, zapíše se do seznamu vah pro pozdější výpis.

## 5) GUI

Jako veřejné rozhraní používáme webovou aplikaci s frameworkem Flask pro Python. Template webové aplikace máme konkrétně v souboru mainPage.html, jehož vzhled po otevření v prohlížeči je na obrázku níže.




Flask aplikaci jako takovou pak máme v souboru main.py.

Aplikaci lze spustit z počítače po stažení celého přiloženého repozitáře. Ke spuštění je potřeba mít stažený python3 interpreter a potřebné knihovny. Pro samotné spuštění aplikace je potřeba zadat ze složky web 2 jednoduché příkazy:

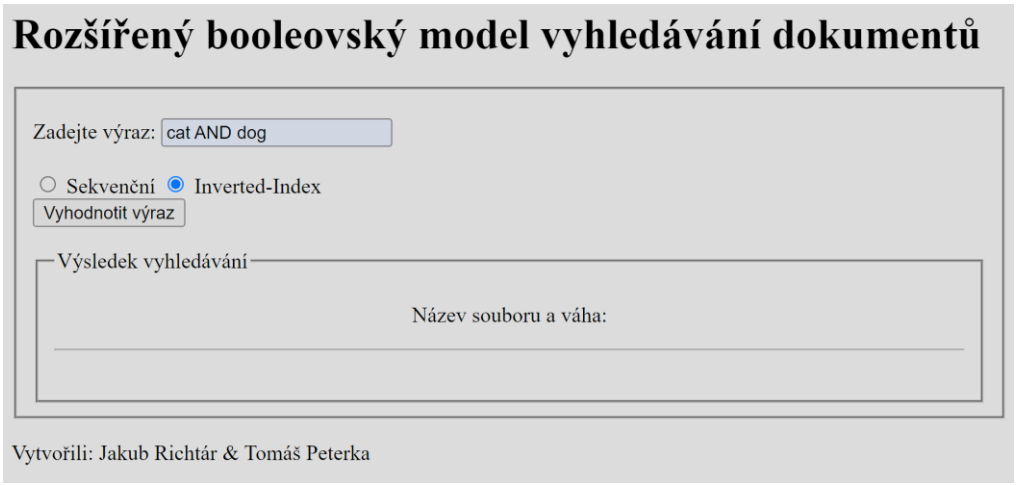
1) **python3 main.py**
2) **python3 -m http.server**

První příkaz spustí samotnou aplikaci, druhý příkaz umožní otevírání textových souborů v prohlížeči.

## Příklady vstupu a výstupu

Příklad 1)

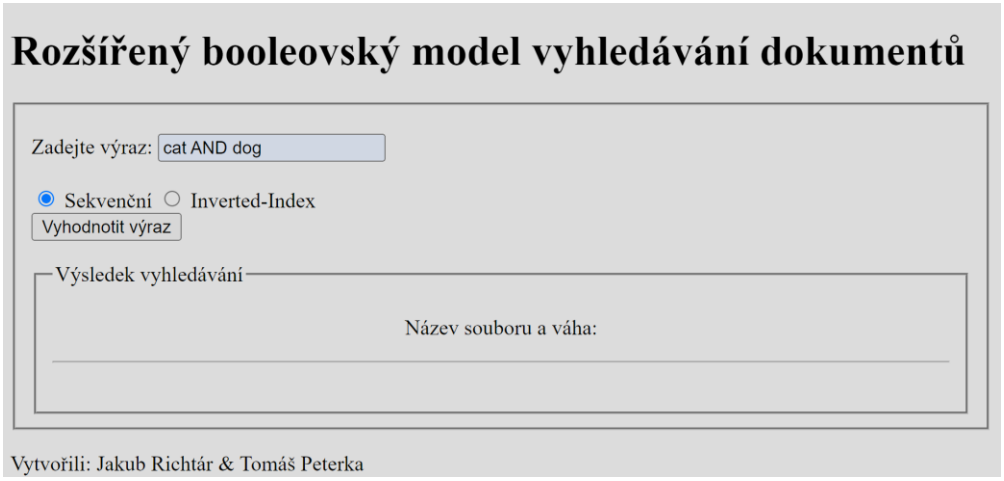
- Vstup:




- Výstup:

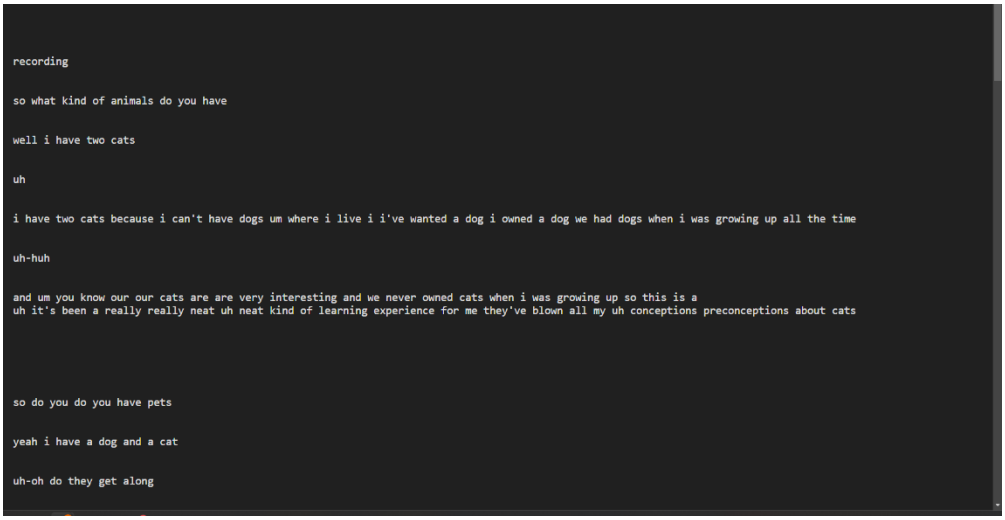

Příklad 2)

- Vstup:



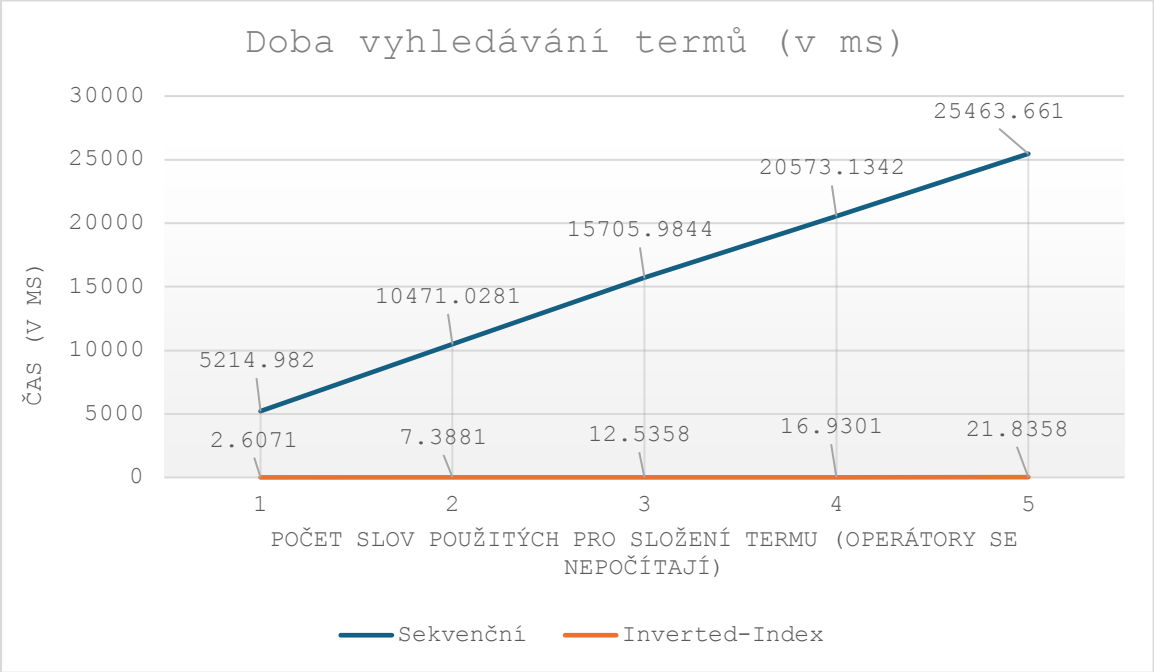

- Výstup:


Pro oba výstupy je možné otevření nalezených souborů kliknutím na hypertextový odkaz názvu souboru:



## Experimentální sekce

Pro tuto sekci jsme si naměřili dobu vyhledávání na základě zadaného výrazu. Postupně jsme zadaný výraz rozšiřovali různými termy a změřené hodnoty poté vykreslili do grafu níže. Měřili jsme hodnoty pro oba způsoby vyhledávání. V grafu je také hezky vidět, jak moc je vyhledávání pomocí Inverted-Indexu rychlejší, oproti sekvenčnímu vyhledávání.




## Diskuze

Naše aplikace je funkční, nicméně by bylo možné ji ještě vylepšit. Například bychom mohli naimplementovat řešení, kde by výsledkem vyhledávání byly soubory, které přesně splňují zadaný výraz. V našem řešení se např. pro výraz “dog AND cat” zobrazí i soubory, které neobsahují term cat. Váha takového souboru je jistě nenulová, takže se soubor vypíše, ale

soubor přesně neodpovídá výrazu, jelikož by měl obsahovat term dog současně s termem cat, což soubor nesplňuje.

## Závěr

Práce nám umožnila hlouběji porozumět tématu rozšířeného boolovského modelu. Ačkoliv se nám tento model na začátku zdál poměrně přímočarý, tak v průběhu práce jsme narazili na několik implementačních problémů, které bylo nutné řešit. Během vypracovávání projektu bylo klíčové zvolit správné technologie a práci si mezi sebe rozdělit na dílčí logické celky. Konečným produktem je funkční implementace rozšířeného boolovského modelu s minimalistickým webovým prostředím.