Boolean Model

Title: Performance Comparison of Inverted Index and Term-Document Matrix in a Boolean Model Search System
Author: Levchenko Maksym

Table of Contents

1. Introduction

This report outlines the development and performance comparison of two different search system architectures: an inverted index and a term-document matrix. Both systems were implemented to handle Boolean query processing, with a focus on evaluating their initialization times and query response times. The primary goal was to determine which system offers a more efficient search mechanism in terms of speed and resource management.

2. Methodology

2.1 Software and Tools

- **Programming Language:** Python 3.10(backend), JavaScript(frontend)
- **Libraries:** Flask, NumPy, NLTK, Pickle, React
- **Development Environment:** Local server setup with Flask providing RESTful API endpoints.
- **Application startup:**

  Backend: cd pythonProject; python3 app.py

  Frontend: cd search-app; npm start

- **Requirments**: Python 3.10 and npm version 6.14.18

2.2 Building the Inverted Index and Term-Document Matrix

- **Document Parsing and Preprocessing:** Each document was tokenized, stopwords were removed, and terms were stemmed using NLTK.
- **Inverted Index Construction:** Created by iterating through preprocessed documents and mapping each unique term to the documents it appears in.

- **Term-Document Matrix Construction:** A matrix was built where rows represent unique terms and columns represent documents, filled with frequency counts of terms in documents.

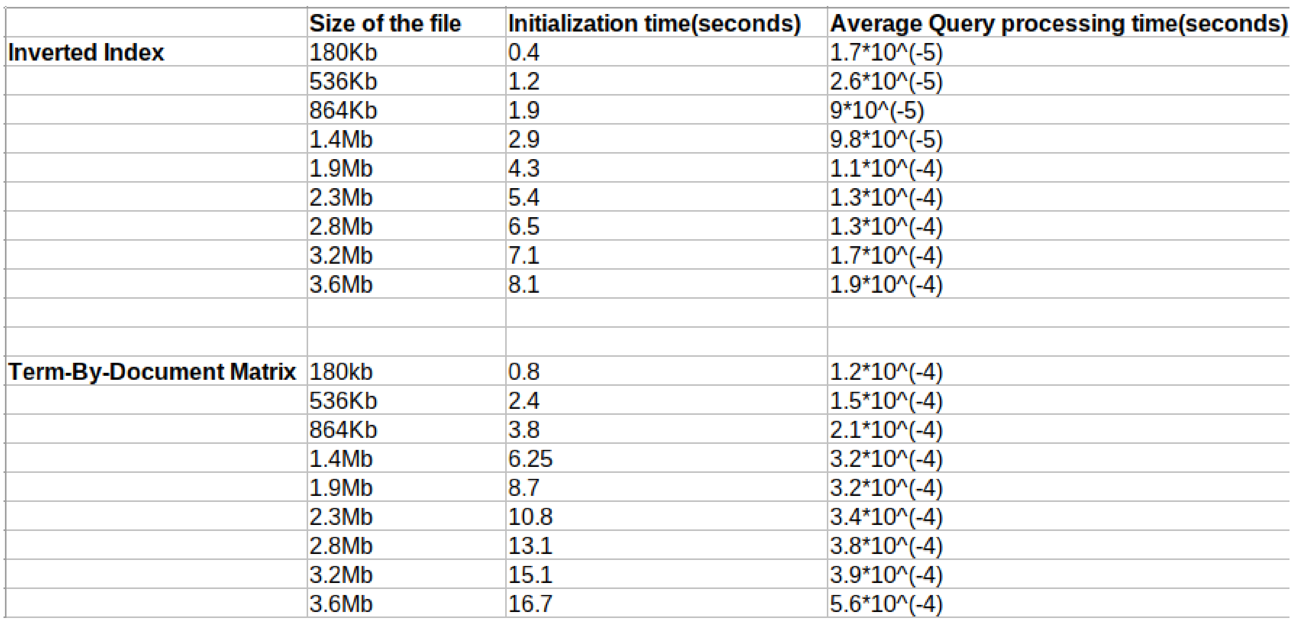
2.3 Query Processing

- **Boolean Query Parsing:** Both systems used a common parser for converting user input into a format suitable for processing (i.e., handling Boolean operators AND, OR, NOT).
- **Query Evaluation:** Employed set operations for the inverted index and matrix manipulations for the term-document matrix to find relevant documents.

3. Results

Results included time measurements for:

- **Initialization:** Time taken to build the inverted index and term-document matrix from scratch.
- **Query Processing:** Average time taken to process different queries across both systems.



	Size of the file	Initialization time(seconds)	Average Query processing time(seconds)
Inverted Index	180Kb	0.4	1.7*10^(-5)
	536Kb	1.2	2.6*10^(-5)
	864Kb	1.9	9*10^(-5)
	1.4Mb	2.9	9.8*10^(-5)
	1.9Mb	4.3	1.1*10^(-4)
	2.3Mb	5.4	1.3*10^(-4)
	2.8Mb	6.5	1.3*10^(-4)
	3.2Mb	7.1	1.7*10^(-4)
	3.6Mb	8.1	1.9*10^(-4)
Term-By-Document Matrix	180kb	0.8	1.2*10^(-4)
	536Kb	2.4	1.5*10^(-4)
	864Kb	3.8	2.1*10^(-4)
	1.4Mb	6.25	3.2*10^(-4)
	1.9Mb	8.7	3.2*10^(-4)
	2.3Mb	10.8	3.4*10^(-4)
	2.8Mb	13.1	3.8*10^(-4)
	3.2Mb	15.1	3.9*10^(-4)
	3.6Mb	16.7	5.6*10^(-4)

4. Discussion

The comparison revealed that the inverted index generally offered faster query processing times, especially for complex queries with multiple Boolean operators, due to more direct and less computationally intensive set operations. Moreover, the inverted index was found to be quicker to initialize.

5. Conclusion

The inverted index provides a more efficient query processing capability in real-world applications where search queries are frequent and data dynamism is less intensive. The term-document matrix could be preferable in environments where the initialization speed is not critical and the queries are relatively simple.