# Sistem Temu Balik Informasi: Implementasi Boolean Retrieval Sederhana dengan Query Optimization

Rhafael Chandra
*Department of Computer Science and Electronics*
*Universitas Gadjah Mada*
Yogyakarta, Indonesia
rhafaelchandra@mail.ugm.ac.id

***Abstrak*—Pencarian informasi yang efektif dan efisien menjadi tantangan utama dalam pengelolaan data besar di era digital. Salah satu metode yang dapat digunakan adalah Boolean Retrieval, yang memungkinkan pencarian dokumen berdasarkan operasi logika. Penelitian ini mengimplementasikan sistem temu balik informasi berbasis Boolean Retrieval sederhana dengan *query optimization*. Hasil menunjukkan bahwa algoritma dengan optimalisasi memberikan peningkatan performa signifikan dalam pemrosesan *query* dibandingkan pendekatan yang tidak menggunakannya, seperti algoritma yang berbasis expression tree, khususnya dalam hal kecepatan.**

***Kata Kunci*—Information Retrieval, Boolean Retrieval, Query Optimization, Inverted Index, Expression Tree, Document Search, Text Processing, Data Preprocessing**

## I. Pendahuluan

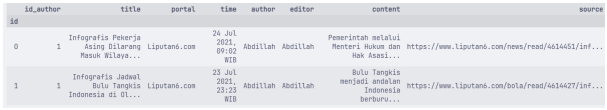
Dalam beberapa dekade terakhir, pertumbuhan data digital yang eksponensial telah menimbulkan tantangan besar dalam pencarian dan pengelolaan informasi. Sistem temu balik informasi adalah kunci untuk mengatasi masalah ini, khususnya dalam konteks dokumen teks. Salah satu teknik paling dasar dalam temu balik informasi adalah Boolean Retrieval, yang menggunakan operator logika sederhana seperti `AND`, `OR`, dan `NOT` untuk menemukan dokumen yang relevan. Meskipun sederhana, metode ini memiliki beberapa keterbatasan dalam hal efisiensi, terutama ketika query menjadi semakin kompleks.

Untuk mengatasi masalah ini, penelitian ini mengusulkan implementasi sistem Boolean Retrieval dengan *query optimization*. Dengan memanfaatkan strategi optimasi, sistem ini diharapkan mampu meningkatkan performa pencarian, baik dari segi kecepatan pemrosesan *query* maupun akurasi hasil. Penelitian ini juga membahas secara mendalam struktur data yang digunakan, seperti *inverted index*, serta membandingkan hasil pemrosesan query pada algoritma optimasi dengan pendekatan berbasis *expression tree*.

## II. Dataset

Dataset yang digunakan adalah kumpulan berita yang teridiri dari 8 kolom dan 14343 baris yang berarti terdapat 14343 dokumen. Dengan detail kolom, yaitu:

- `id_author`: id penulis berita
- `title` : judul berita
- `portal` : portal berita
- `time` : waktu berita diposting
- `author` : penulis berita
- `editor` : editor berita
- `content` : isi berita
- `source` : sumber berita (url)



| id | id_author | title | portal | time | author | editor | content | source |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | Infografis Pekerja Asing Dilarang Masuk Wilaya... | Liputan6.com | 24 Jul 2021, 09:02 WIB | Abdillah | Abdillah | Pemerintah melalui Menteri Hukum dan Ham Asasi... | https://www.liputan6.com/news/read/4614451/inf... |
| 1 | 1 | Infografis Jadwal Bulu Tangkis Indonesia di Ol... | Liputan6.com | 23 Jul 2021, 23:23 WIB | Abdillah | Abdillah | Bulu Tangkis menjadi andalan Indonesia berburu... | https://www.liputan6.com/bola/read/4614427/inf... |

Gambar 1: Sampel Dataset

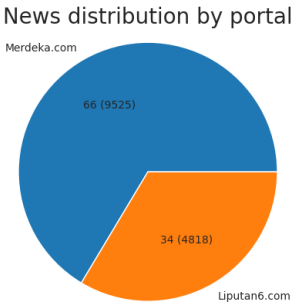
Terdapat beberapa kolom yang tidak memiliki nilai yaitu kolom `content` sebanyak 9 baris dan `source` sebanyak 150 baris. Selain itu, terdapat beberapa data duplikat pada dataset


Gambar 2: Pola Missing Value

sebanyak 122 baris yang merupakan 0.85% dari dataset. Terdapat juga 1 baris dengan konten yang sama dengan detail kolom lain yang berbeda.

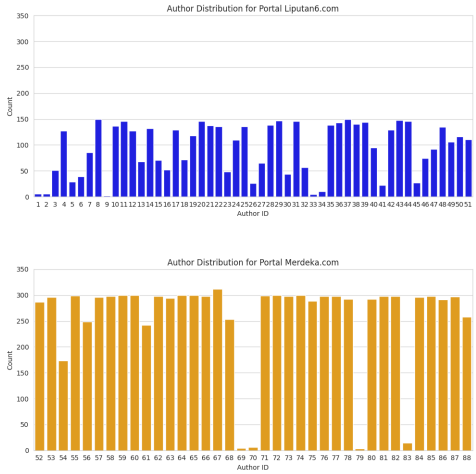
Terdapat dua portal berbeda pada data yang diberikan, yaitu `Merdeka.com` dan `Liputan6.com` dengan persebarannya secara berturut-turut adalah 66% dan 34% dari keseluruhan data.

Masing-masing portal, memiliki penulis yang berbeda juga. Dapat dilihat pada Gambar 4, sesama penulis berita dari portal `Merdeka.com` cenderung memiliki jumlah berita yang sama dan jumlah berita mereka cenderung lebih banyak dibandingkan dengan portal `Liputan6.com`, sedangkan sesama



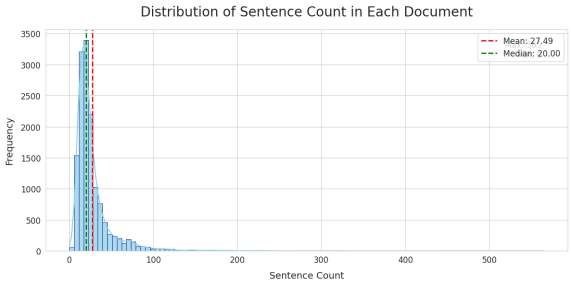

Gambar 3: Distribusi Portal

penulis dari portal `Liputan6.com` memiliki jumlah berita yang fluktuatif.




Gambar 4: Distribusi Penulis Berita Berdasarkan Portal

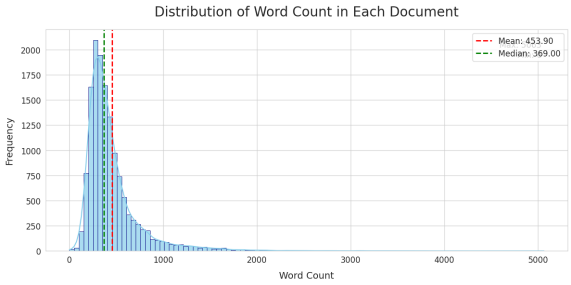
Persebaran kalimat pada setiap dokumen tidaklah merata, *right-skewed*, yang dapat dilihat pada Gambar 5. Mean kalimat pada setiap dokumen adalah 27.49 kalimat, sedangkan median-nya adalah 20 kalimat. Persebaran kata pada setiap dokumen



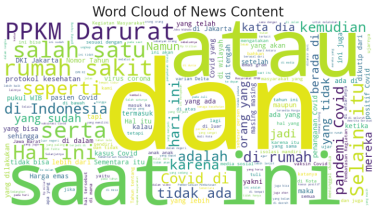

Gambar 5: Distribusi Kalimat

juga tidaklah merata, *right-skewed*, yang dapat dilihat pada Gambar 6. Mean kata pada dokumen adalah 453.90 kata, sedangkan mediannya adalah 369 kata.

Kata yang mendominasi dokumen adalah kata-kata *stop-words* yang dapat dilihat pada Gambar 7.




Gambar 6: Distribusi Kata




Gambar 7: Word Cloud

## III. Metodologi

Pada penelitian ini, terdapat tiga tahapan utama, yaitu prapemrosesan data, pembuatan *inverted index dan universe*, dan pemrosesan *query*. Pada tahap prapemrossan data, pemilihan kolom akan dilakukan dan pembersihan data terhadap kolom tersebut juga akan dilakukan. Pada tahap pembuatan *inverted index*, setiap kata/*term* yang ada di tiap dokumen akan disimpan pada sebuah *dictionary*. Pada tahap pemrosesan *query*, algoritma yang dapat menerima *query* dan mengembalikan dokumen yang sesuai dengan *query* tersebut akan dibuat.

### A. Prapemrosesan Data

Prapemrosesan data dilakukan pada kolom `content`. Tahapan yang ada di bagian ini mencakup penghapusan *missing values*, penghapusan *duplicated data*, dan tokenisasi dokumen.

*1) Penghapusan missing values: Missing values* akan dihapus dari data yang nantinya akan diproses karena jumlahnya sangat kecil jika dibandingkan dengan keseluruhan dataset, yaitu 9 dari 14343.

*2) Penghapusan duplicated data: Duplicated data* akan dihapus dari data yang nantinya akan diproses karena jumlahnya sangat kecil jika dibandingkan dengan keseluruhan dataset, yaitu 123 dari 14343.

*3) Tokenisasi:* Tokenisasi akan diterapkan pada dataset yang sudah dibersihkan. Proses ini akan memisahkan kata-kata menggunakan *regular expression* yang telah ditetapkan. Lebih detail, penelitian ini menggunakan function dari `NLTK`, yaitu `word_tokenize`. Hasil dari proses ini adalah array yang berisi string untuk setiap kata yang telah ditokenisasi.

### B. Pembuatan Inverted Index dan Universe

Inverted index akan dibuat yang merupakan dictionary dari setiap kata yang ada pada keseluruhan dokumen yang akan menunjuk pada `id` setiap dokumen. Universe akan dibuat

untuk menampung semua `id` dokumen yang ada. Representasi kumpulan dokumen yang digunakan adalah array yang elemennya adalah integer yang menyatakan jumlah dokumen dan array yang elemennya adalah `id` dokumen, seperti `[num_docs, [id_docs...]]`.

*1) Inverted Index:* Setiap kata dalam dokumen akan diiterasi dan disimpan dalam sebuah *dictionary*. Proses ini akan menghasilkan *dictionary* yang memiliki kunci kata unik. Nilai kunci *dictionary* tersebut akan diisi dengan representasi kumpulan dokumen yang mencakup dokumen yang mencakup kata dari kunci tersebut.

*2) Universe:* Universe direpresentasikan dengan representasi keseluruhan dokumen yang ada, perlu diperhatikan bahwa universe juga menampung `id` dokumen dari dataset yang belum dibersihkan.

### C. Pemrosesan Query

Terdapat dua pendekatan pada tahapan ini, yaitu pemrosesan berbasis *tree* dan pemrosesan yang dioptimasi. Input yang diterima adalah *query* dalam format string yang memenuhi aturan `AND`, `NOT`, `OR`, `(`, dan `)` yang dipisahkan oleh spasi, contoh:

- DITERIMA: `luhut.pandjaitan AND menmbah AND ( NOT Gonzalez ) AND Colombia`
- DITOLAK: `luhut.pandjaitan AND menmbah AND(NOT Gonzalez) AND Colombia`

Output yang diberikan adalah representasi dokumen. Prioritas pemrosesan *query* mengikuti aturan, dari tertinggi hingga terendah:

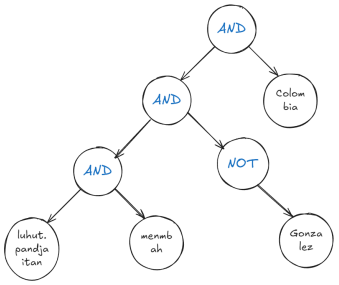
- `(` dan `)`
- `NOT`
- `AND`
- `OR`

*1) Tree Based Processing:* Tahap yang dilakukan pada pemrosesan ini mencakup *parsing*, pembuatan *tree*, dan *tree traversal*.

- *Parsing* : Pemisahan item pada *query* akan dilakukan berdasarkan spasi dan disimpan dalam bentuk representasi dokumen.
- Pembuatan *Tree*: Expression tree akan dibuat berdasarkan *query*, dengan leaf node-nya adalah *term*, yaitu kata yang ada pada *inverted index*, dan node selain itu adalah operator, yang mencakup `AND`, `NOT`, `OR`, `(`, dan `)`. Contoh dapat dilihat pada Gambar 8.
- *Tree traversal*: *Expression* tree yang telah dibuat akan dilalui secara rekursif untuk menghasilkan representasi dokumen akhir.

*2) Query Optimization:* Tahap yang dilakukan pada pemrosesan ini mencakup *parsing*, pemrosesan kurung, pemrosesan item tanpa kurung, pemrosesan item dengan operator yang sama.

- *Parsing* : Pemisahan item pada *query* akan dilakukan berdasarkan spasi dan disimpan dalam bentuk representasi dokumen.




Gambar 8: Expression Tree

- Pemrosesan kurung : Semua item yang ada di dalam `(` dan `)` akan diproses lebih lanjut. Tahap ini akan dilakukan terus menerus hingga tersisa satu representasi dokumen.
- Pemrosesan item tanpa kurung: Item yang didapatkan dari tahapan sebelumnya, akan diproses berdasarkan prioritas yang ada. `NOT` akan diproses terlebih dahulu untuk semua *term* yang diberikan. Semua *term* pada `AND` yang sekuensial akan diberikan pada tahap selanjutnya. Semua *term* pada `OR` yang sekuensial akan diberikan pada tahap selanjutnya. Hasil yang didapatkan dari tahap tersebut akan menggantikan posisi *term* paling kiri dan sisa item *query* akan digeser sehingga proses ini akan dilakukan secara berulang hingga item yang tersisa adalah satu representasi dokumen. Hasil ini kemudian akan dikembalikan pada tahap sebelumnya.
- Pemrosesan item dengan operator yang sama: Item yang didapatkan dari tahapan sebelumnya akan diproses berdasarkan ukuran dokumennya. Setiap dua representasi dokumen dengan jumlah dokumen terkecil akan diproses terlebih dulu hingga tersisa satu representasi dokumen. Hasilnya ini kemudian akan dikembalikan pada tahap sebelumnya.

## IV. Hasil dan Pembahasan

Program akan dijalankan pada Google Colab.

### A. Prapemrosesan

*1) Penghapusan missing values:* Hasil dari tahapan ini menyisakan 14334 dokumen, dengan 9 data dihapus.

*2) Penghapusan duplicated data:* Hasil dari tahapan ini menyisakan 14211 dokumen, dengan 123 data dihapus.

*3) Tokenisasi:* Waktu yang dibutuhkan akan dihitung bersamaan dengan pembuatan *inverted index* dan *universe*.

### B. Pembuatan Inverted index dan Universe

Waktu yang dibutuhkan dengan detail:

- Tokenisasi
- Pembuatan *inverted index*
- Pembuatan *universe*

adalah `52.864378994 s`.

TABEL I: Hasil Pemrosesan *Query*

| N Items | Time (ms) | Time Opt (ms) | Improv. (%) |
|---|---|---|---|
| 5 | 3.432115 | 2.608371 | 24.00 |
| 10 | 10.226173 | 7.036737 | 31.19 |
| 15 | 9.705648 | 8.350556 | 13.96 |
| 20 | 24.112310 | 20.853683 | 13.51 |
| 25 | 14.162325 | 9.971587 | 29.59 |
| 30 | 43.103553 | 31.323429 | 27.33 |

### C. *Pemrosesan Query*

*Query generator* akan dibuat untuk membuat *query* secara otomatis dengan parameter yang diberikan adalah aproksimasi panjang item dengan aturan operator yang telah ditentukan. Akan dibuat 6 kelompok kueri, yaitu kueri dengan panjang 5, 10, 15, 20, 25, dan 30. Masing masing kelompok tersebut akan memiliki 10 *query*. Pemrosesan kueri akan diterapkan pada $6 \times 10$ *query*. Hasil pemrosesan *query* dapat dilihat pada Tabel I dan Gambar 9.


Gambar 9: Perbandingan Waktu Pemrosesan *Query* dan Peningkatannya

## V. KESIMPULAN DAN SARAN

Berdasarkan hasil pemrosesan *query*, algoritma yang menggunakan *query optimization* akan berjalan lebih cepat dibandingkan dengan algoritma yang tidak menggunakannya, seperti pendekatan *expression tree* pada berbagai panjang item *query*.

Saran penulis untuk perkembangan lebih lanjut lagi adalah pemrosesan data tingkat lanjut, seperti *case folding*, *stemming*, penggantian bahasa gaul atau slang, dapat dilakukan untuk memberikan hasil yang lebih optimal.

## LAMPIRAN

Google Colab: https://s.id/Colab_STBI2_498550