ANALISA PERBANDINGAN QUERY PENCARIAN MENGGUNAKAN FUNGSI MATCH-AGAINST PADA MYSQL DENGAN TABEL KAMUS

Zudha Pratama[1]
Yans Safarid Hudha[2]
M Lukman Prayoghi[3]
[1,2,3]Magister Teknik Informatika STMIK AMIKOM Yogyakarta
E-mail: [1]zudhapratama@gmail.com, [2]yans.0862@students.amikom.ac.id,
[3]Prayoghi.indo@gmail.com

Diterima: 4 Januari 2018/ Disetujui : 22 Januari 2018

ABSTRACT

Virtually all database-related systems provide search features. Starting from a complex search engine like google to a relatively simple example of search features on a digital library page. A good search engine is capable of delivering fast, accurate, and fault-tolerant results. Speed may be affected by server device capabilities and complex algorithm combinations.The form of the condition condition used in the search query generally uses LIKE for partial search, REGEXP for multi key search, and MATCH-AGAINST for multi key search with fulltext index. However, these functions are not sufficient to perform a search selection on a slightly wrong key or rather fault tolerance that is still not good. So researchers do an analysis if one of the search function is combined with a dictionary table.Table dictionary as a comparator key to find a more appropriate key if key wrong key. But on the other hand the addition of the comparison process is estimated to have a weakness to the processing time. Researchers assume if the weakness can be overcome if the ability of the server is improved.

Keywords : *Query optimization, fulltext index, searching, match-against, fault tolerance*

ABSTRAK

Hampir semua sistem yang berhubungan dengan basis data menyediakan fitur pencarian. Mulai dari mesin pencarian kompleks seperti google hingga yang relatif sederhana misal fitur pencarian pada sebuah halaman digital library. Mesin pencarian yang baik adalah yang mampu memberikan hasil yang cepat, akurat, dan memiliki toleransi kesalahan yang baik. Kecepatan mungkin dipengaruhi oleh kemampuan perangkat server dan kombinasi algoritma yang kompleks.Bentuk fungsi kondisi yang digunakan pada query pencarian umumnya menggunakan LIKE untuk pencarian parsial, REGEXP untuk pencarian multi key, dan MATCH-AGAINST untuk pencarian multi key dengan index fulltext. Namun fungsi-fungsi tersebut belum cukup untuk melakukan seleksi pencarian pada key yang sedikit salah atau lebih tepatnya toleransi kesalahan yang masih kurang baik. Maka peneliti melakukan analisa jika salah fungsi pencarian tersebut dikombinasikan dengan sebuah tabel kamus.Tabel kamus tersebut sebagai pembanding key untuk mencarikan alternatif key yang lebih tepat jika key salah ketik. Namun disisi lain penambahan proses pembandingan tersebut diperkirakan memiliki kelemahan terhadap waktu proses. Peneliti berasumsi jika kelemahan tersebut dapat ditanggulangi jika kemampuan server ditingkatkan.

Kata Kunci *: optimasi query, index fulltext, pencarian, match-against, toleransi kesalahan*

1. PENDAHULUAN

Fitur pencarian dapat kita temukan hampir di setiap sistem pengolahan data yang berbasis database. Fitur tersebut memudahkan pengguna untuk mencari data yang dikehendakinya. Pencarian data pada dasarnya merupakan proses filter yang menyeleksi himpunan data menjadi himpunan data yang lebih kecil dengan kriteria tertentu. Kriteria tersebut berkaitan dengan kesesuaian unsur teks substring penyusun teks dengan kata kunci pencarian. Atau lebih tepatnya adalah kemiripan kata kunci dengan himpunan data yang dicari.

Mesin pencarian yang baik adalah yang mampu memberikan hasil yang cepat, akurat, dan memiliki toleransi kesalahan yang baik. Kecepatan mungkin dipengaruhi oleh kemampuan perangkat server. Kemudian hasil akurat dipengaruhi algoritma yang digunakan pada saat proses query. Dan toleransi kesalahan seperti kurang spasi atau kata kunci yang terbalik juga dipengaruhi algoritma yang digunakan, namun dengan algoritma yang lebih kompleks.

Untuk merumuskan penelitian ini kami menggunakan beberapa literatur review dari penelitian terdahulu yang memiliki keterkaitan bahasan dengan penelitian ini, berikut literatur yang berhasil kami kumpulkan:

Menurut C. Gyorodi et al (2010) proses dalam *fulltext-search*, mesin pencari memeriksa semua kata di setiap dokumen tersimpan dengan mencoba mencocokkan kata pencarian yang diberikan oleh pengguna. Saat sebuah query menangani pencarian pada sejumlah dokumen, fungsi tersebut memindai isi dokumen dengan setiap query. Cara ini disebut *serial scanning* yang merupakan unsur dasar saat melakukan fungsi pencarian. Query *full-text* pada MySQL mengembalikan baris hasil pencarian sesuai relevansinya, MySQL menghitung sebuah angka untuk menentukan relevansinya di dalam teks. Sehingga hasilnya memiliki relevansi yang sesuai dengan kata kunci pencarian dari pengguna [1].

Kemudian Manish Sharma (2013) menjelaskan tujuan utama dari sistem pencarian informasi adalah untuk menemukan informasi yang relevan atau dokumen yang memenuhi kebutuhan informasi pengguna. Biasanya sistem pencarian menerapkan 3 proses yakni : *indexing* ,dokumen diwakili dalam bentuk konten yang dirangkum. *Filtering*, semua *stopword* dan kata-kata umum dibuang. Proses terakhir dan yang utama *searching*, pemindaian dokumen yang sesuai dengan kebutuhan pengguna [2].

Selanjutnya Akram Roshdi (2015) menyimpulkan bahwa sistem pencarian informasi adalah proses pencarian dan pengambilan informasi berbasis pengetahuan dari kumpulan dokumen. Wujud penerapan teknik pencarian informasi pada himpunan dokumen teks dalam skala besar adalah mesin pencarian. Contohnya yang paling dikenal adalah mesin pencari web, dan banyak pencarian lain, seperti: pencarian desktop, pencarian enterprise, pencarian federasi, pencarian mobile, dan pencarian social [3].

Pendapat lain, Aruleba (2016) menyatakan jika sistem pencarian informasi dirancang untuk mengambil dokumen atau informasi yang dibutuhkan oleh kumpulan pengguna. Terutama ditargetkan untuk membuat informasi yang tepat, tersedia bagi pengguna yang tepat pada waktu yang tepat. Berupa proses menampilkan, mencari, dan memanipulasi kumpulan data teks elektronik yang besar dan mungkin tidak terstruktur. merupakan tanggapan atas serangkaian perintah *query*[4].

Pada pendapat diatas menyebutkan jika kumpulan data yang akan dicari berupa teks elektronik yang besar dan mungkin tidak terstruktur. Maka agar konten lebih mudah diakses, data perlu disimpan dalam format terstruktur yang disebut indeks. Menurut Mabayoje (2013) bentuk indeks yang paling sederhana adalah daftar diurutkan dari semua kata dan frase yang ditemukan dalam konten dokumen yang telah diambil. Kata-kata dan ungkapan akan disimpan dalam urutan abjad disertai dengan sumber dan peringkat atau popularitas mereka [5].

Untuk teknik pencarian data, Laxmi et al (2014) menjelaskan ada 2 cara untuk melakukan pencocokan dalam *searching*: pertama adalah *Calling User-Defined Functions (UDF).* Kita bisa menambahkan fungsi ke dalam database untuk memverifikasi apakah sebuah record berisi kata kunci query. Dan yang kedua adalah menggunakan predikat *LIKE*. *Database* menyediakan predikat *LIKE* untuk memungkinkan pengguna melakukan pencocokan string. Kita bisa menggunakan predikat *LIKE* untuk memeriksa apakah sebuah *record* berisi kata kunci query atau tidak. Predikat *LIKE* tersebut diuji dan dikombinasikan dengan pencarian multi kata kunci. Hasilnya memberikan kesimpulan jika query pencarian multi kata kunci dengan dikombinasikan dengan predikat *LIKE* ternyata sangat

membantu dalam proses pencarian [6]. Penjelasan multi kata kunci dalam literatur kelima ini memberikan ide kepada kami untuk melakukan proses pencarian multi kata kunci dengan penggabungan dua perintah query dengan *UNION* untuk pencarian yang menggunakan kata kunci asli (kata kunci yang diketikkan pengguna) dengan kata kunci alternatif.

Kemudian penggunaan tabel kamus kata kunci untuk menampung kemungkinan kata kunci alternatif, peneliti memperoleh ide tersebut dari penggunaan tabel kamus pada proses *stopword removal*. Untuk mengenai table kamus telah sedikit disinggung dalam sebuah penelitian yang berjudul *Learning Vector Quantization untuk Klasifikasi Abstrak Tesis* yang ditulis oleh Fajar Rohman Hariri dkk (2015). Proses *stopword removal* tersebut merupakan salah satu bagian dari proses *pre-proccessing text-mining* pada proses klasifikasi. Yang terdiri *pre-proccessing* tersebut terdiri *parsing*, *stopword removal dan stemming* [7]. *Parsing* merupakan pemecahan kalimat menjadi kata dan *stemming* adalah proses merubah kata berimbuhan menjadi kata dasar.

Sebenarnya jika *stemming* juga digunakan maka dapat meningkatkan toleransi kesalahan kata kunci karena proses pencarian dapat mengantisipasi jika terdapat ada kata berimbuhan dalam kata kunci dan terdapat kata baku dalam data yang dicari atau sebaliknya. Namun karena fokus pembahasan hanya pada pengaruh tabel kamus terhadap hasil yang memberikan toleransi kesalahan kata kunci serta pertimbangan jika seluruh tahap pada *pre-processing* digunakan akan memberikan beban proses yang lebih lama maka pada penelitian ini belum menerapkan seluruh tahapan pada *pre-processing* tersebut.

Namun fokus utama yang di bahas dalam penelitian ini adalah perbandingan hasil pencarian dengan fungsi Match-Against dan fungsi Match-Against yang dikombinasikan dengan tabel kamus. Kemudian dilakukan analisa apakah waktu prosesnya lebih lama jika menggunakan tabel kamus serta bagaimanakah pengaruhnya terhadap toleransi kesalahan kata kunci. Analisa tersebut diperoleh dari pengujian hasil implementasi algoritma pencarian pada bahasa pemrograman database atau bahasa SQL (*Structured Query Language*), dalam hal ini peneliti menggunakan database MySQL. Kemudian untuk fungsi Match-Against yang digunakan tersebut merupakan fungsi bawaan yang sudah ada pada MySQL. Peneliti hanya membuat algoritma untuk pengkombinasiaanya dengan tabel kamus. Serta menguji apakah toleransi kesalahan pada pencarian dengan tabel kamus lebih baik dibanding tanpa menggunakan tabel kamus dan bagaimanakah pengaruhnya terhadap waktu proses.

Dalam proses analisa perbandingan yang akan dilakukan peneliti menggunakan beberapa dasar teori antara lain:

1.1. SQL

Menurut raharjo (2011), SQL yaitu kependekan dari Structured Query Language, yang merupakan bahasa atau kumpulan perintah standar yang digunakan untuk berkomunikasi dengan database[8]. Menurut ichwan (2011), kegunaan bahasa SQL yaitu: (a) membangun basis data, (b) menjalankan query terhadap basis data, (c) melakukan penambahan, pengurangan, perubahan terhadap data yang ada[9]. Sedangkan dalam penelitian ini SQL digunakan sebagai alat pengujian implementasi algoritma dengan memanfaatkan fungsi Match-Against pada Mysql, pada field yang memilki index bertipe fulltext yang dikombinasikan dengan tabel kamus.

1.2. Fulltext Search

Dalam buku elektronik dari ilmukomputer.com, Didik Setiawan mengartikan Fulltext Searching adalah fungsionalitas yang terdapat pada database (dalam hal ini database MySQL) yang memungkinkan user untuk melakukan pencarian tertentu dalam tabel. Dengan cara melakukan perbandingan string[10]. Dapat menjadi pilihan untuk melakukan fungsi pencarian pada database MySQL. Untuk melakukan pencarian fultext adalah menggunakan fungsi Match-Against. Syarat untuk dapat menjalankan fungsi tersebut adalah dengan menambahkan index dengan tipe fulltext pada field yang di jadikan acuan pencarian.

1.3. Tabel Kamus

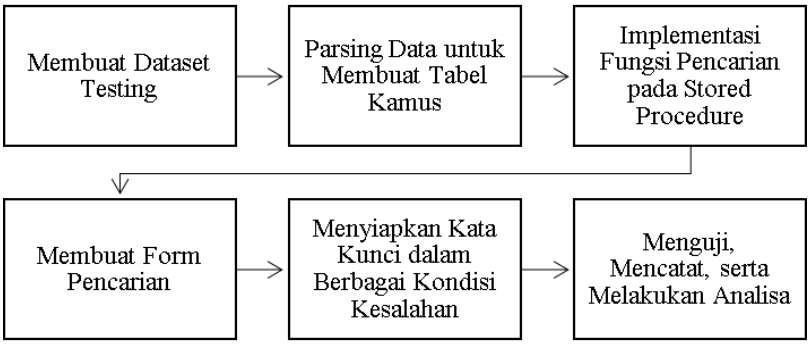
Sebenarnya tabel kamus merupakan hanya alat bantu saja, bukan merupakan aturan atau metode yang memiliki dasar teori yang jelas. Belum pernah dijelaskan secara rinci mengenai tabel

kamus. Peneliti menggunakan tabel kamus karena terinspirasi dari penggunaannya pada proses stemming. Stemming merupakan proses konversi kata berimbuhan menjadi kata baku yang sering digunakan pada proses *pre-proccessing* dalam *text-mining*.

2. METODE PENELITIAN

Proses pengujian diawali dengan membuat dataset testing. Data tersebut hanya berisi kode dan judul saja. Dalam hal ini penulis menggunakan data proyek pribadi yang sudah ada yakni data judul film. Kemudian membuat data kamus yang diperoleh dari hasil parsing judul. Judul diparsing berdasarkan spasi menjadi kata kemudian jika kata hasil parsing belum ada dalam tabel kamus maka kata tersebut disimpan.

Berikutnya menyiapkan fungsi yang dimplementasikan pada stored procedure dan sebuah halaman web yang menampilkan inputan kata kunci dan tabel data hasil pencarian. Kemudian dilakukan pengujian dengan memasukkan kata kunci dengan kondisi normal dan kata kunci yang sengaja dibuat salah dengan beberapa kondisi tertentu (tanpa spasi, posisi kata terbalik, posisi kata terbalik dengan tanpa spasi, kata kurang ketik, dan kata kelebihan ketik). Dicatat hasilnya kemudian dibandingkan. Untuk lebih jelasnya proses penelitian silakan lihat diagram berikut :



Gambar 1. *Urutan Proses Penelitian*

3. HASIL DAN PEMBAHASAN

Pengujian perbandingan algoritma pencarian pada penelitian ini dimulai dari menentukan algoritma pencarian itu sendiri. Algoritma pencarian terdiri dari penggabungan algoritma pencarian dengan kata kunci asli dan algoritma pencarian dengan kata kunci alternatif (lihat gambar 2). Penggunaan kata kunci alternatif untuk memberikan hasil pencarian lain yang memiliki kemiripan. Alternatif itu sendiri merupakan nilai *sub-string* (potongan kata) dari kata kunci utama yang sama dengan kata yang tersimpan dalam tabel kamus.

Gambar 2. *Algoritma Pencarian (proses pencarian dengan kata kunci asli dan alternatif)*

Algoritma yang telah ditentukan di atas kemudian dikonversi menjadi bahasa pemrograman database. Berupa sebuah *stored procedure* MySQL yang nantinya dipanggil pada halaman pencarian. S*tored procedure* dipilih karena bersifat modular, berupa potongan perintah tersendiri dalam database server yang dapat dipanggil melalui berbagai bahasa pemrograman lain, baik berbentuk desktop maupun web. Sehingga untuk kedepannya bisa dikembangkan ke berbagai *platform software*. Untuk potongan baris kode *stored procedure* dapat di lihat pada gambar 4.

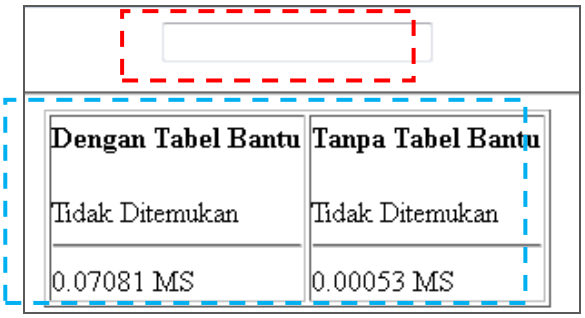
```
CREATE PROCEDURE `search1`(sKey VARCHAR(255))
BEGIN
SELECT judul from film WHERE MATCH judul AGAINST(sKey);
END;
```

Gambar 3. *Potongan Kode SQL Implementasi Algoritma Pencarian Tanpa Tabel Kamus*

```
CREATE PROCEDURE `search2`(sKey VARCHAR(255))
BEGIN
DECLARE sKeyAlternate LONGTEXT;
SET @tmp:='';
SET sKeyAlternate = (SELECT max(t.s) FROM
(SELECT kamus.kata,@tmp:=concat(@tmp,kamus.kata,'|')as s
FROM kamus WHERE POSITION(kamus.kata in sKey))t);
SET sKeyAlternate = LEFT(sKeyAlternate,LENGTH(sKeyAlternate)-1);
SELECT judul from film WHERE MATCH judul AGAINST(sKey)
UNION
SELECT judul from film WHERE MATCH judul AGAINST(sKeyAlternate);
END;
```

Gambar 4. *Potongan Kode SQL Implementasi Algoritma Pencarian Dengan Tabel Kamus*

Stored procedure hasil implementasi diatas selanjutnya dipanggil dari sebuah halaman pencarian. Halaman tersebut terdiri isian kata kunci dan hasil pencarian. Hasil pencarian terdiri dari dua bagian, yakni hasil dengan tabel bantu dan tanpa tabel bantu. Masing-masing hasil pencarian disertai waktu proses pencarian. Lihat gambar 5.

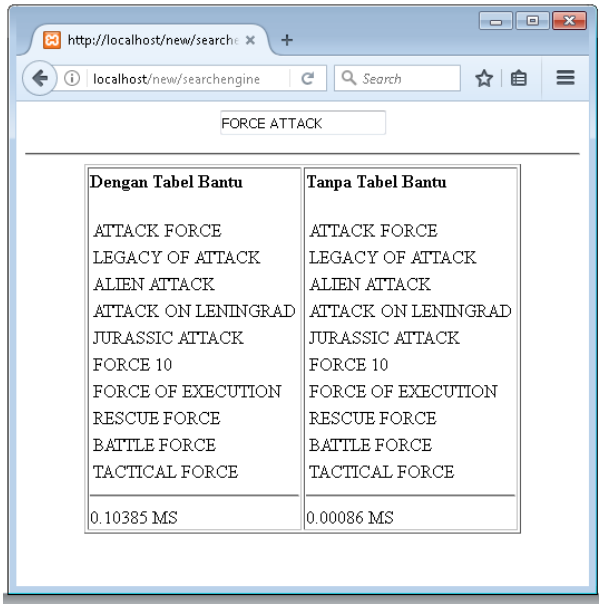


Gambar 5. *Tampilan Halaman Uji Algoritma Pencarian*

3.1. Hasil Pengujian

3.1.1. Pengujian Dengan Kata Kunci Normal

Kata kunci normal disini adalah kata kunci yang memang sebelumnya diketahui sesuai dengan salah satu data. Berikut hasil pengujian kami dengan memberikan kata kunci normal pada halaman uji.

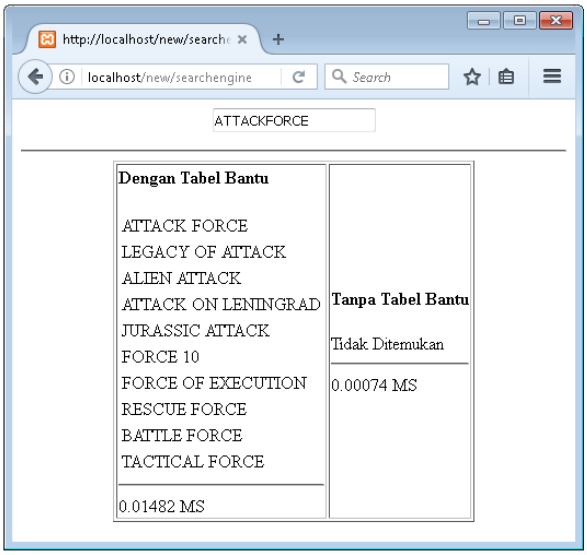


Gambar 6. *Hasil Pengujian Dengan Kata Kunci Normal*

Pengujian diatas menunjukan jika pencarian menggunakan kata kunci yang normal (benar) maka kedua algoritma pencarian menampilkan hasil pencarian yang sama namun dengan waktu proses yang berbeda. Pencarian yang menggunakan tabel kamus membutuhkan waktu proses yang lebih lama. Pencarian yang sesuai dengan kata kunci terletak pada baris paling atas dan dibawahnya terdapat hasil yang mirip dengan salah satu kata pada kata kunci.

3.1.2. Pengujian Dengan Kata Kunci Tanpa Spasi

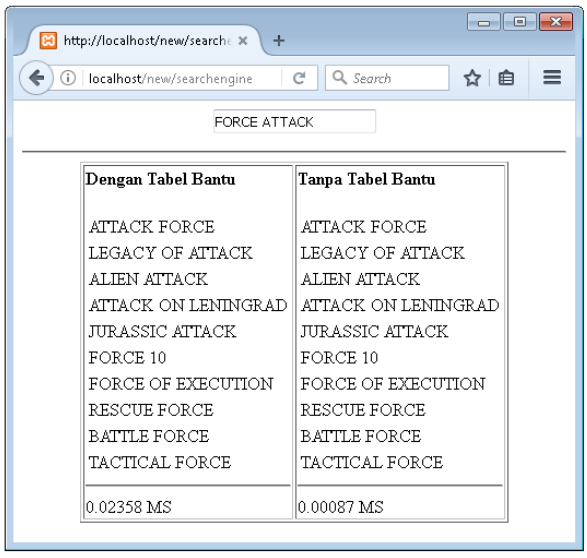
Kata kunci tanpa spasi disini adalah kata kunci yang sama seperti kata kunci pada pengujian pertama namun sengaja dibuat tanpa spasi. Berikut hasil pengujian kami dengan memberikan kata kunci tanpa spasi pada halaman uji.



Gambar 7. *Hasil Pengujian Dengan Kata Kunci Tanpa Spasi*

Pengujian diatas menunjukan jika pencarian menggunakan kata kunci tanpa spasi, maka kedua algoritma pencarian menampilkan hasil pencarian yang berbeda. Terlihat hasil tidak ditemukan pada pencarian tanpa spasi. Dan dari segi waktu proses menunjukkan hasil yang mirip seperti pengujian pertama, pencarian yang menggunakan tabel kamus membutuhkan waktu proses yang lebih lama.

3.1.3. Pengujian Dengan Posisi Kata Kunci Terbalik



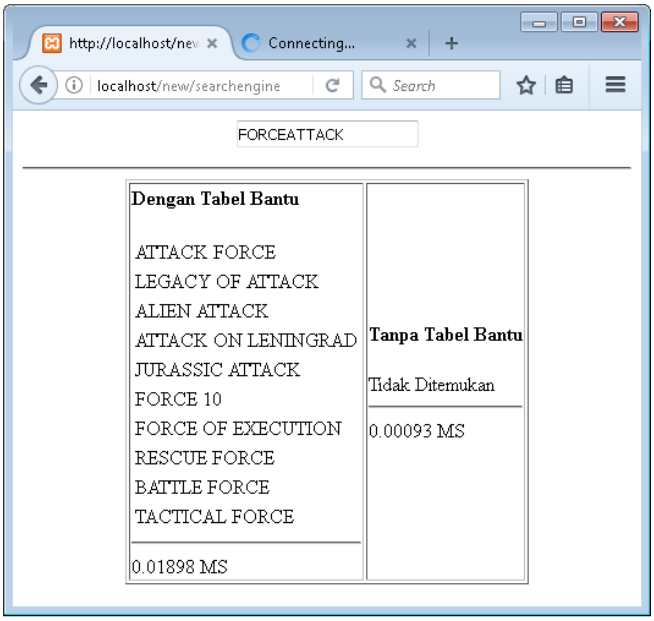
Gambar 8. *Hasil Pengujian Dengan Posisi Kata Kunci Terbalik*

Maksud Kata kunci terbalik adalah kata kunci yang sama seperti kata kunci pada pengujian pertama namun sengaja dibuat terbalik posisinya. Kejadian ini sering terjadi saat kita kurang hafal atau ragu saat menuliskan kata kunci.

Pengujian diatas menunjukan jika pencarian menggunakan kata kunci dengan posisi kata terbalik maka kedua algoritma pencarian menampilkan hasil pencarian yang sama. Untuk pengujian ke-tiga ini jika dilihat waktu prosesnya hasilnya sama seperti pengujian sebelumnya. Pencarian yang menggunakan tabel kamus membutuhkan waktu proses yang lebih lama.

3.1.4. Pengujian Dengan Posisi Kata Kunci Terbalik dan Tanpa Spasi

Ini merupakan penggabungan kondisi pengujian dua dan tiga. Berikut hasil pengujian tersebut.

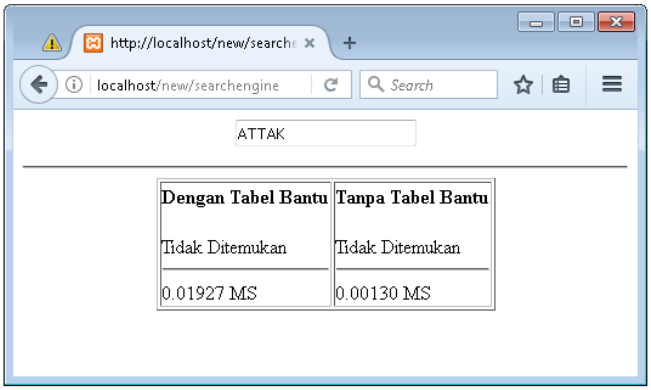


Gambar 9. *Hasil Pengujian Dengan Posisi Kata Kunci Terbalik dan Tanpa Spasi*

Pengujian diatas menunjukan jika pencarian menggunakan kata kunci yang normal (benar) maka kedua algoritma pencarian menampilkan hasil pencarian yang tidak sama. Pencarian yang menggunakan tabel kamus membutuhkan waktu proses yang lebih lama. Sama seperti pengujian-pengujian sebelumnya dan hanya berbeda angka saja.

3.1.5. Pengujian Dengan Kata Kunci Kurang Ketik

Kata kunci kurang ketik adalah kondisi yang tidak sengaja sering terjadi saat kita terburu-buru atau memang kurang hafal dengan apa yang kita cari. Berikut hasil pengujian kata kunci kurang ketik.

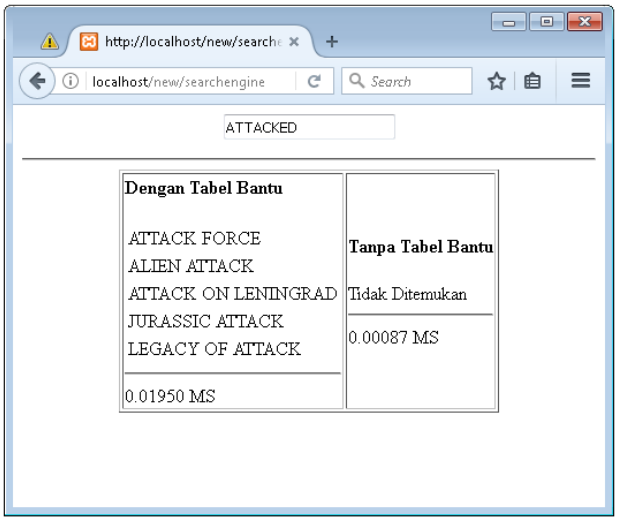


Gambar 10. *Hasil Pengujian Dengan Kata Kunci Kurang Ketik*

Pengujian diatas menunjukan jika pencarian menggunakan kata kunci kurang ketik maka kedua algoritma pencarian menampilkan hasil pencarian yang sama-sama bernilai kosong. Dengan waktu proses seperti semua pengujian sebelumnya jika menggunakan tabel kamus membutuhkan waktu proses yang lebih lama.

3.1.6. Pengujian Dengan Kata Kunci Kelebihan Ketik

Sama seperti pengujian kelima ini juga kondisi yang tidak sengaja sering terjadi saat kita terburu-buru atau memang kurang hafal dengan apa yang kita cari. Berikut hasil pengujian kata kunci kelebihan ketik.



Gambar 11. *Tampilan Halaman Implementasi Algoritma Pencarian Menggunakan Tabel Kamus*

3.2. Analisa Hasil Pengujian

Proses analisa dilakukan dengan membuat tabel pernadingan hasil pencarian. Hasil-hasil pencarian diatas dijadikan tabel perbandingan. Sehingga dengan mudah kita merumuskan hasil analisa perbandingan kedua algoritma. Dan pada tahap akhir kita mendapatkan suatu simpulan yang terukur dan berdasarkan fakta pengujian. Bukan dari hasil kira-kira atau dugaan tanpa data.

Tabel 1. Perbandingan Pencarian Dengan Tabel Kamus Dan Tanpa Tabel Kamus

No. Uji	Kondisi Kata Kunci	Waktu Proses		Kesesuaian Hasil Pencarian	
		Dengan Tabel Kamus	Tanpa Tabel Kamus	Dengan Tabel Kamus	Tanpa Tabel Kamus
1	Normal	0,10385	0,00086	Sesuai	Sesuai
2	Tanpa spasi	0,01482	0,00074	Sesuai	Tdk Sesuai
3	Posisi kata terbalik	0,02358	0,00087	Sesuai	Sesuai
4	Posisi kata terbalik dan tanpa spasi	0,01898	0,00093	Sesuai	Tdk Sesuai
5	Kata kurang ketik	0,01927	0,00130	Tdk Sesuai	Tdk Sesuai
6	Kata kelebihan ketik	0,01950	0,00087	Sesuai	Tdk Sesuai

3.3. Hasil Analisa

3.3.1. Semua nomor uji menunjukkan jika penambahan proses pembandingan pada tabel kamus dapat memberikan beban waktu proses yang lebih lama.

3.3.2. Penggunaaan fungsi Match-Against pada algoritma pencarian memberikan hasil yang sesuai, jika kondisi kata kunci normal atau terbalik saja. Selain itu pencarian bernilai null (kosong).

3.3.3. Kombinasi fungsi Match-Against dikombinasikan dengan tabel kamus memberikan hasil yang lebih baik, hampir pada semua kondisi kata kunci dapat menghasilkan alternatif hasil pencarian. Hanya mengalami kegagalan pencarian pada saat kondisi kata kunci kurang ketik.

4. KESIMPULAN

Pengujian algoritma pencarian menggunakan fungsi Match-Against dengan tabel kamus serta analisa terhadap hasil pengujian tersebut peneliti mendapatkan kesimpulan sebagai berikut:

4.1. Dari segi toleransi kesalahan pencarian menggunakan Match-Against dengan tabel kamus memberikan hasil pencarian yang lebih baik dibanding pencarian yang hanya menggunakan fungsi Match-Against saja.

4.2. Hasil pencarian menggunakan Match-Against dengan tabel kamus memiliki toleransi kesalahan penulisan kata kunci pada semua kondisi kesalahan yang diujikan kecuali jika kata kunci kurang ketik.

4.3. Penambahan proses pembandingan *substring* kata kunci pada tabel kamus dapat memberikan beban waktu proses yang lebih lama.

4.4. Hasil penelitian diatas juga sesuai dengan tulisan C. Bhagya Laxmi,dkk mengenai query pencarian multi kata kunci yang ternyata sangat membantu dalam pencarian. Pencarian dengan multi kata kunci memberikan alternatif hasil pencarian.

5. SARAN

Pengujian algoritma menggunakan fungsi Match-Against dengan tabel kamus merupakan ide yang salah satunya berasal dari penggunaan tabel kamus pada stemming yang merupakan bagian dari *pre-proccessing text-mining* pada *string-matching*. Masih banyak yang di manfaatkan dari tahap-tahap *pre-proccessing text-mining* yakni stemming dan stopword removal. Proses tersebut dapat ditambahkan pada proses pencarian untuk antisipasi data yang memiliki kata berimbuhan dan mengandung stopword, agar toleransi kesalahan kata kunci pada proses pencarian lebih baik.

Sehingga untuk yang berminat melakukan penelitian lanjutan mengenai algoritma pencarian dapat mengambil salah satu tahap *pre-proccessing text-mining* pada *string-matching* untuk dilakukan pengujian dan analisa kelebihan serta kekurangannya terhadap algoritma pencarian yang sudah ada.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Ibu Dra.Kusrini,M.Kom selaku dosen kami yang telah memberi pengarahan dan dukungan motivasi terhadap penelitian ini.

DAFTAR PUSTAKA

[1] Gyorodi, C., R. Gyorodi, G. Pecherle, dan G. M. Cornea, 2010, Full-Text Search Engine using MySQL, *International Journal of Computers, Communications & Control*, No.5, Vol.5, hal. 735-743.

[2] Sharma, Manish dan Rahul Patel, 2013, A Survey on Information Retrieval Models, Techniques And Applications, *International Journal of Emerging Technology and Advanced Engineering*, No.11, Vol.3, hal. 542-545.

[3] Roshdi, Akram dan Akram Roohparvar, 2015, Review: Information Retrieval Techniques and Applications, *International Journal of Computer Networks and Communications Security*, No.9, Vol.3, hal. 373–377.

[4] Aruleba, K.D., Akomolafe, D.T. and Afeni, B., 2016, A Full Text Retrieval System in a Digital Library Environment, *Intelligent Information Management*, 8, hal 1-8.

[5] Mabayoje, Modinat. A. dan Olawale S. Adebayo, 2013, A Full-text Website Search Engine Powered by Lucene and The Depth First Search Algorithm, *International Journal Computer Network and Information Security,* 3, hal 1-12

[6] Laxmi, C. Bhagya, Bhaludra Raveendranadh Singh dan Moligi Sangeetha, 2014, Supporting Search-As-You-Type Using SQL in Databases, *International Journal of Computer Trends and Technology.* No.5, Vol.16, hal 185-188.

[7] Hariri, Fajar Rohman, Ema Utami dan Armadyah Amborowati., 2015, Learning Vector Quantization untuk Klasifikasi Abstrak Tesis, *Creative Information Technology Journal,* No.2, Vol.2, hal128-143.
[8] Raharjo, Budi. 2011. *Membuat Database Menggunakan MySql.* Informatika, Bandung.
[9] Ichwan, M. 2011. *Pemograman Basis Data Delphi 7 Dan MySql*. Informatika, Bandung.
[10] Setiawan, Didik, 2013, *MySQL Full-Text Searching*, http://ilmukomputer.org/wp-content/uploads/2013/09/di2k-MySQL-Full-text-Searching.pdf, diakses tanggal 18 Mei 17.