Nusantara Journal of Multidisciplinary Science
Vol. 1 No. 1, Agustus 2023
Hal 145-156
E-ISSN : 3024-8752
P-ISSN : 3024-8744
Site : https://jurnal.intekom.id/index.php/njms

Implementasi Algoritma *Levenstein Distance* Dan Algoritma *Knutt Morris Pratt* Dalam *Fitur Word Completion* Pada *Search Engine*

Ryan Dhika Priyatna[1]
[1] Program Studi Pendidikan Teknik Informatika, Sekolah Tinggi Keguruan Dan Ilmu Pendidikan Al Maksum, Stabat, Indonesia

Article Info

Article history:

Received August 10, 2023
Revised August 20, 2023
Accepted August 30, 2023

Keywords:

Algoritma
Word Completion
Search Engine
String

Keywords:

Algorithm
Word Completion
Search Engine
String

ABSTRAK

Search engine adalah program komputer yang dirancang untuk membantu seseorang menemukan file-file yang disimpan dalam komputer. Dengan adanya mesin pencari setiap orang dapat dengan mudah memperoleh informasi yang diinginkan. Perkembangan teknologi internet mendorong munculnya *fitur* dan inovasi terbaru untuk meningkatkan pengalaman dan kemudahan pengguna dalam menjelajahi dunia maya. Fitur tersebut dinamakan *Word Completion*. *Fitur word completion* memberikan pengalaman baru dan kemudahan bagi pengguna untuk memperoleh informasi . Hasil dari penelitian ini dengan menggunakan dua algoritma adalah bahwa algoritma *Levenstein Distance* digunakan sebagai koreksi kesalahan kata dengan cara substitusi, eliminasi, penambahan dan *algoritma Knuth Morris Pratt* mencari kata dengan cara menggeser *string* kesebelah kanan sampai kata tersebut dinyatakan cocok. Penelitian ini sangat penting karena dapat membantu mempermudah untuk melakukan pencarian kata yang dicari tanpa mengetikkan seluruh kata, sehinggga kata yang dicari akan muncul sesuai yang ada didalam database.

ABSTRACT

Computer programmes called search engines are made to make it easier for users to locate files on a computer. Anyone may quickly find the information they need with search engines. The latest features and innovations to enhance users' experience and convenience of cyberspace exploration have emerged as a result of the advancements in internet technology. Word Completion is the name of this function. Users can acquire information more easily and enjoy a fresh experience using the word completion tool. The Levenstein Distance algorithm is used to fix word errors by substitution, removal, and addition, while the Knuth Morris Pratt algorithm looks for words by shifting the string to the right until the word is deemed a match. These are the findings of this research employing two algorithms.

The significance of this research lies in its potential to facilitate word searches without requiring the whole word, thereby ensuring that the terms you seek appear exactly as they do in the database. (ukuran 10 pt)

NJMS : Nusantara Journal of Multidisciplinary Science **E-ISSN : 3024-8752**
Vol. 1, No. 1, Agustus 2023, Hal 145-156 **P-ISSN : 3024-8744**

Corresponding Author:

Ryan Dhika Priyatna
Program Studi Pendidikan Teknik Informatika, STKIP Al Maksum
Langkat, Indonesia
Email: ryandhikapriyatna@gmail.com

1. PENDAHULUAN

Perkembangan teknologi internet mendorong munculnya *fitur* dan inovasi terbaru untuk meningkatkan pengalaman dan kemudahan pengguna dalam menjelajahi dunia maya. Salah satunya *fitur* tersebut adalah word completion merupakan *fitur* yang di implementasikan pada *web browser* dan mesin pencari yang memungkinkan *web browser* atau mesin pencari untuk memberikan saran pencarian ketika baru beberapa kata diketikkan dalam kolom pencarian *address bar.* Ketika menggunakan *web browser* untuk berjelajah di dunia maya maka pada kolom *address bar* saat mengetikkan beberapa huruf/kata, *web browser* tersebut memberikan saran dari apa yang diketik, atau sama halnya ketika menggunakan mesin pencari untuk mencari suatu informasi, ketika baru mengetikan beberapa huruf/kata pada kolom pencarian, mesin pencari tersebut memberikan saran pencarian yang terkait dari apa yang diketikan. Ketika memilih saran tersebut, *web browser* atau mesin pencari akan langsung melengkapi kata kunci pencarian sesuai dengan saran yang dipilih. Fitur tersebut dinamakan *word completion*. *Fitur word completion* memberikan pengalaman baru dan kemudahan bagi pengguna ketika berjelajah di internet. faktor yang mempengaruhi keberagaman pada *word completion* dan analisis semantik pada mesin pencari .

Search engine atau mesin pencari adalah program komputer yang dirancang untuk membantu seseorang menemukan file-file yang disimpan dalam komputer. Dengan adanya mesin pencari setiap orang dapat dengan mudah memperoleh informasi yang diinginkan. Dengan mengetikkan kata yang ingin dicaripada mesin pencari maka seluruh informasi yang diinginkan akan ditampilkan. Namun sejumlah penelitian terhadap mesin pencari menyimpulkan bahwa rata-rata kesalahan dalam pengetikan kata yang dicari yang dilakukan oleh pengguna cukup tinggi . *Word completion*, adalah fitur yang disediakan oleh banyak *web browser*, surel, antarmuka mesin pencari, *source code editor*, *tools* pada *query* database, pengolah kata (word processor), dan *interpreter* pada *command line word completion* juga terdapat dan sudah terintegrasi dalam teks editor yang umum digunakan. Kegunaan dari *fitur word completion* ini adalah menampilkan perkiraan kata atau frase yang akan dimasukkan tanpa harus mengetikkan keseluruhan kata. *Auto Correct* atau disebut juga *Auto Text* adalah suatu *fitur* yang biasanya dapat ditemukan di berbagai macam aplikasi seperti pemroses kata dan program untuk produk *Apple*, seperti *iPod,iPhone* dan *iPad. Auto Correct* ini berfungsi untuk mempersingkat waktu pengetikan sebuah kata .

2. METODE

Algoritma *Levenshtein* merupakan algoritma yang digunakan untuk mencari jumlah operasi *string* yang paling sedikit untuk mentransformasikan suatu *string* menjadi *string* . yang Algoritma ini digunakan dalam pencarian *string* dengan pendekatan perkiraan (*Approximate String Matching*) . *Levehenstein Distance* dibuat oleh Vladimir Levehenstein pada tahun 1965. Perhitungan *edit distance* dihitung oleh matriks yang digunakan unuk menghitung jumlah perbedaan *string* antara dua *string*. *Levenshtein Distance* jarak antara dua *string* didefinisikan sebagai jumlah minimum suntingan yang diperlukan untuk mengubah satu string ke yang lain, dengan diijinkan mengedit operasi yang penyisipan, penghapusan, atau penggantian karakter

E-ISSN : 3024-8752
P-ISSN : 3024-8744

tunggal . Perhitungan jarak antara dua *string* ditentukan dari dua jumlah minimum operasi perubahan untuk membuat string A mejadi *string* B .

Algoritma ini berjalan mulai dari pojok kiri atas sebuah *array* dua dimensi yang telah diisi sejumlah *string* awal dan *string* target dan diberikan nilai *cost*. Nilai *cost* pada ujung bawah kanan menjadi nilai edit *distance* yang mengambarkan jumlah perbedaan dua *string*. Berikut ini adalah table matriks perhitungan edit distance :

Tabel 1. Matriks Perhitungan *Edit distance*

		w	o	r	d
	0	1	2	3	4
w	0	1	1	2	3
r	2	1	2	1	2
d	3	2	1	2	1

Contoh dari perhitungan *Levenshtein* mengunakan 2 *string* yang berbeda kemudian dihitung *edit distance* nya pada tabel 1. Dapat dilihat dari perhitungan *edit distance* antara 2 *string* ‘wrd’ dan ‘word’ adalah 1. Pengecekan dimuai dari iterasi awal dari kedua *string* kemudian dilakukan operasi penambahan, penyisipan dan penghapusan. Nilai *edit distance* nya yaitu pada ujung kanan matriks. Hanya ada satu proses penyisipan yaitu penyisipan karakter ‘o’ pada *string* ‘wrd’ sehingga menjadi ‘word’. Pada kasus pengecekan ejaan proses perhitungan ini dilakukan sejumlah kata yang ada pada basis data. Tentu saja untuk saran yang terbaik dibutuhkan daftar kata berbahasa Indonesia yang lengkap. Sehingga kata yang disarankan bisa mendekati yang diharapkan oleh pengguna.
Perhitungan jarak antara dua *string* ditentukan dari dua jumlah minimum operasi perubahan untuk membuat string A menjadi *string* B .

C. *Algoritma Knuth-Morris-Pratt*

Knuth Morris Pratt (KMP) dikembangkan oleh D. E. Knuth, bersama dengan J. H. Morris dan V. R.Pratt. Untuk pencarian *string* dengan menggunakan algoritma Knuth Morris Pratt , setiap kali ditemukan ketidakcocokan pattern dengan teks, maka *pattern* akan digeser satu karakter ke kanan.Algoritma ini membandingkan pola dengan teks dari kiri ke kanan [7]. pada algoritma *Knuth Morris-Pratt*, kita memelihara informasi yang digunakan untuk melakukan jumlah pergeseran. Algoritma KMP digunakan untuk bekerja pada arsitektur yang mendukung string paralel ukuran yang lebih besar.

Algoritma menggunakan informasi tersebut untuk membuat pergeseran yang lebih jauh dan melakukan evaluasi waktu . Secara sistematis, langkah-langkah yang dilakukan algoritma *Knuth-Morris-Pratt* pada saat mencocokkan *string* :

1. Algoritma *Knuth-Morris-Pratt* mulai mencocokkan *pattern* pada awal teks.
2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:

1. Karakter di pattern dan di teks yang dibandingkan tidak cocok (*mismatch*).
2. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
3. Algoritma kemudian menggeser *pattern* berdasarkan tabel, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

E-ISSN : 3024-8752
P-ISSN : 3024-8744

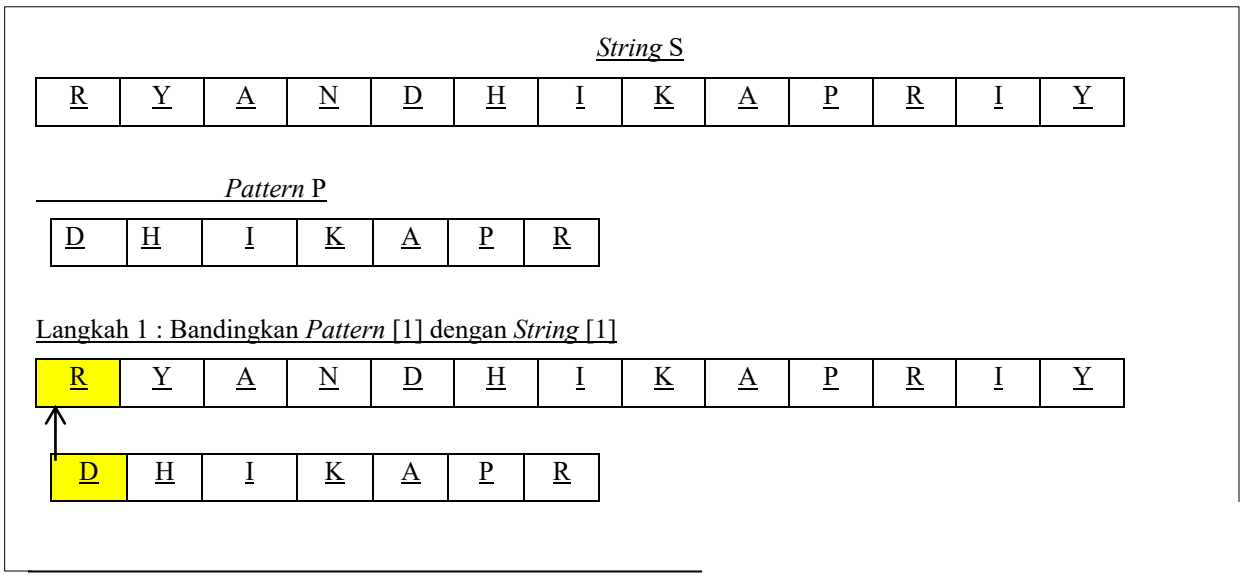
Pencocokan karakter dari kiri ke kanan mencari *prefix* terpanjang dari P[0..j-1] yang juga merupakan *suffix* dari P [1..j-1], untuk menghindari pergeseran yang tidak perlu. Hasil dari pencarian *prefix* terpanjang disimpan dalam tabel yang disebut juga sebagai *failure function*. Misalkan panjang *string* yang telah diperiksa dan cocok = *n* dan nilai dari *failure function* adalah M, maka dilakukan pergeseran sebanyak (*n-m*).
Contoh penggunaan algoritma *string matching KMP*, yaitu :

Teks : RYAN DHIKAPRIY

Pattern : DHIKAPR

Cara kerja :

Berikut ini adalah gambar cara kerja algoritma *Levenstein Distance*



Gambar 1. Pergeseran String Knuth Morris Pratt

Algoritma *Levenshtein Distance* terbukti dapat menyelesaikan beberapa permasalahan dalam penelitian ilmiah, penelitian yang pernah dilakukan yang berkaitan dengan algoritma *Levenshtein Distance*, diantaranya yaitu [6]. Algoritma *Levenshtein Distance* pada fitur *Autocomplete* pada Aplikasi Katalog Perpustakaan.

Layanan *autocomplete (Word completion)* telah banyak digunakan pada penelitian terdahulu. layanan *autocomplete* juga pernah diterapkan pada *Smart Phones* [5] dengan menggunakan kombinasi algoritma *Brute Force, Boyer-Moore* dan *Knuth-Morris Pratt*.

Penelitian yang pernah dilakukan yang berkaitan dengan algoritma *Knuth-Morris Pratt* yaitu [3] Aplikasi Algoritma Pencarian *String Knuth-Morris-Pratt* dalam Permainan *Word Search* kesimpulan nya adalah algoritma KMP dapat menyimpan sebuah informasi yang digunakan untuk melakukan jumlah pergeseran, sehingga algoritma ini melakukan pergeseran lebih jauh tidak hanya bergeser satu karakter.

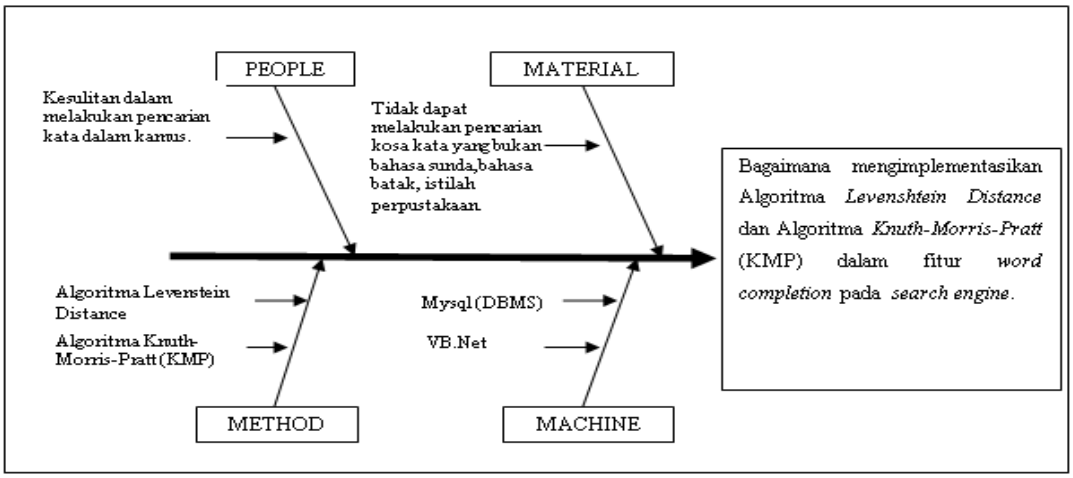
Penelitian tentang *string matching* juga pernah dilakukan oleh [9] dengan judul *String Matching Algorithms and their Applicability in various Applications.* kesimpulan nya adalah jarak antara dua string didefinisikan sebagai jumlah minimum yang diperlukan untuk

mengubah satu string ke yang lain, dengan diijinkan mengedit operasi penyisipan, penghapusan, atau penggantian karakter tunggal.

D. Analisis Dan Perancangan Sistem

1. Diagram Ishikawa

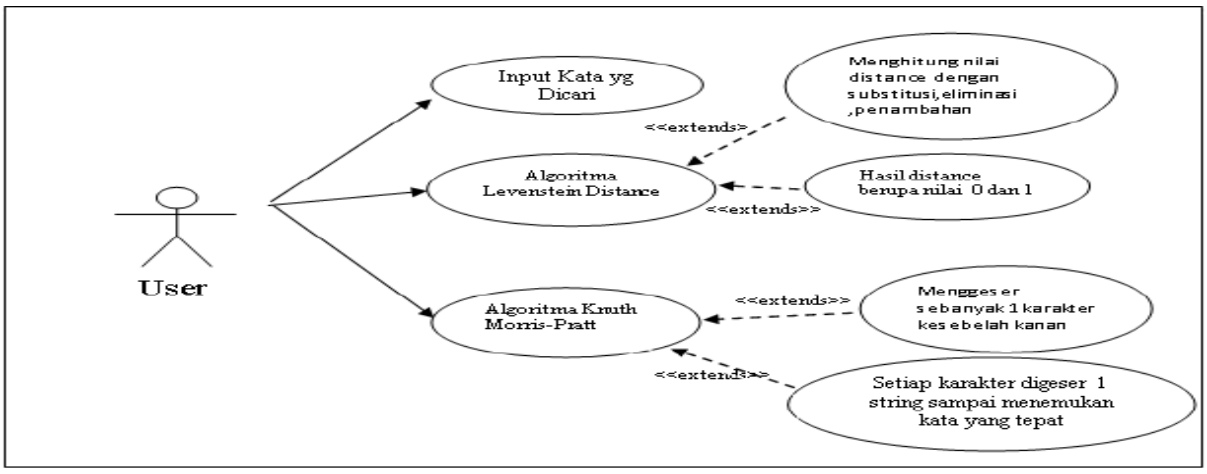
Diagram *Ishikawa* yang dapat digunakan untuk menganalisis masalah. Bagian kepala atau segiempat yang berada di sebelah kanan merupakan masalah. Sementara di pada bagian tulang merupakan penyebab. Gambar 3 menunjukkan diagram sistem algoritma *levenstein distance* dan algoritma *knuth morris pratt*.



Gambar 2. Diagram Sistem Algoritma LD dan KMP

2. Use-Case Diagram

Use case merupakan fungsionalitas dari suatu sistem, sehingga *customer* atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun. *Use case* berperan menggambarkan interaksi antar komponen-komponen yang berperan dalam sistem yang akan dirancang. Gambar 4 menunjukkan use case diagram .

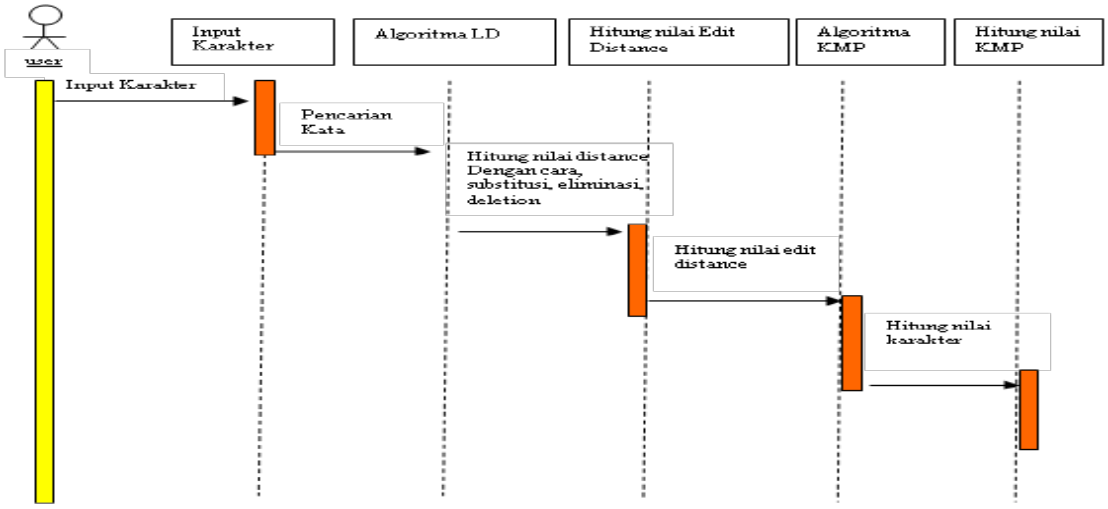


Gambar 3. Use Case Diagram

E-ISSN : 3024-8752
P-ISSN : 3024-8744

3. Sequence Diagram

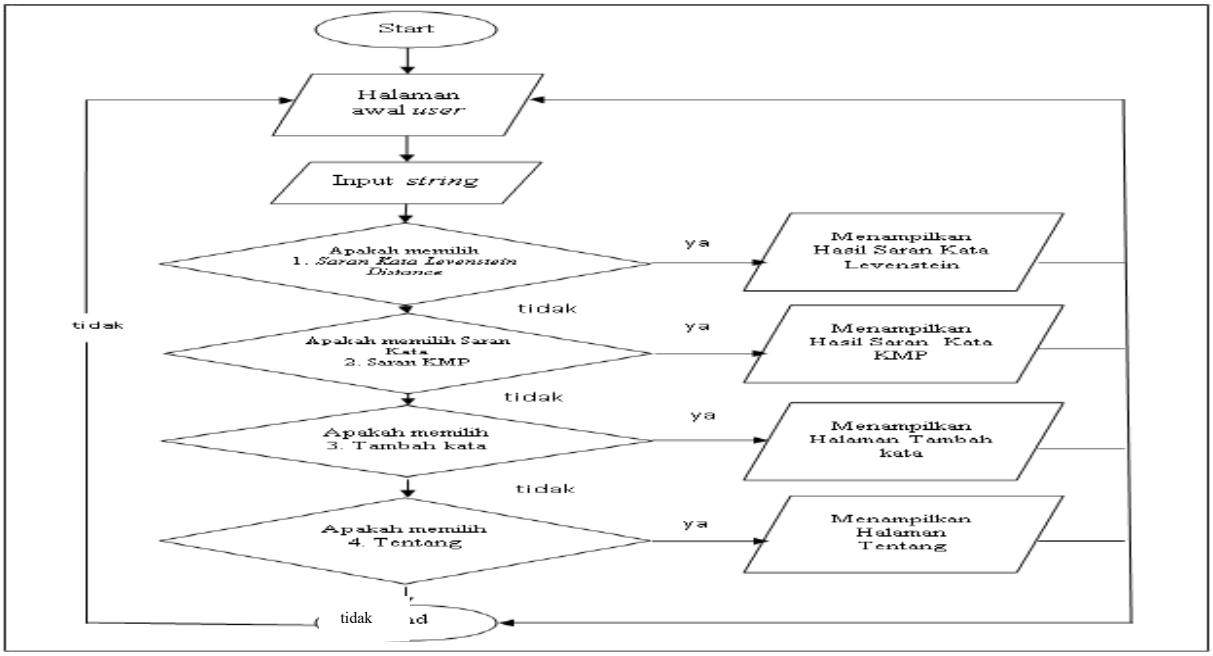
Sequence diagram merupakan diagram yang menggambarkan interaksi antar objek dan menjelaskan bagaimana suatu operasi dilakukan. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh objek. Dalam sistem yang akan dibangun, interaksi dilakukan antara pengguna dan system. Gambar 4 menunjukkan *sequence diagram.*



Gambar 4. Sequence Diagram

E. Flowchart Sistem

Pada gambar 6. Menggambarkan flowchart system algoritma levenstein distance dan knuth morris pratt .



Gambar 5. Flowchart Sistem

F. Analisis Data

Data yang digunakan pada sistem ini berdasarkan data kata yang ada dan telah disimpan pada *database,* dimana *database* tersebut berisi kata yang digunakan dalam bahasa sunda. Pada

tabel 2 akan diberikan beberapa data kata yang akan digunakan sebagai database pada algoritma *Levenstein Distance* dan Algoritma *Knuth Morris* . Berikut ini merupakan table bahasa sunda.

Tabel 2.Sampel Data Kamus Bahasa Sunda

No.	Kamus Bahasa Sunda
1.	abah
2.	abdi
3.	ameh
4.	aben
5.	ablag
6.	aber

Dapat dilihat pada tabel 2 bahwa data yang telah disimpan dalam database adalah sebanyak 6 data kata, dimana data tersebut akan dijadikan sebagai *database* yang implementasikan pencariannya dalam *search engine* dengan algoritma *Levenstein Distance* dan algorima *Knuth-Morris-Pratt.*

G. Perancangan Antar Muka (Interface)

Antarmuka **(***Interface*) berguna untuk mempermudah pengguna ketika mengakses sebuah aplikasi, antarmuka sistem juga merupakan suatu alur komunikasi antara *user* dengan sistem. Antarmuka dapat disebut juga sebagai wajah dari suatu aplikasi, sehingga ketika seorang *user* atau pengguna pertama kali mengakses suatu aplikasi, maka bagian pertama yang akan muncul adalah antarmuka (*interface*).

H. Cara Kerja Algoritma Levenstein Distance

1. Operasi pengubahan karakter
   Operasi pengubahan karakter merupakan operasi menukar sebuah karakter dengan karakter lain, contohnya penulis menuliskan *string* ‘yamg’ menjadi ‘yang’. Dalam kasus ini karakter ‘m’ diganti menjadi huruf ‘n’.
2. Operasi penambahan karakter
   Operasi penambahan karakter berarti menambakan karakter kedalam suatu *string.* Contoh nya *string* ‘kepad’ menjadi ‘kepada’, dilakukan penambahan *string* ‘a’ diakhir *string*. Penambahan karakter ini tidak hanya diakhir kata, namun bisa diawal atau ditengah *string.*
3. Operasi penghapusan karakter
   Operasi penghapusan karakter dilakukan untuk menghilangkan karakter pada suatu *string.* Contohnya *string* ‘barur’ karakter terakhir dihilangkan

E.. Cara Kerja Algoritma Knuth-Morris-Pratt

1. Menentukan *Pattern* dan Teks

Algoritma *Knuth-Morris-Pratt* adalah algoritma pencocokan *string* yang juga tediri dari dua komponen utama, yaitu *pattern* dan teks.

2. Proses Fitur *Word Completion*

Setelah *pattern* dan teks terbentuk maka proses selanjutnya adalah melakukan pencocokan karakter. Berbeda dengan algoritma *Levenstein Distance*, algoritma *Knuth Morris-Pratt* ini

dapat memelihara informasi yang digunakan untuk melakukan jumlah pergeseran. Algoritma *Knuth-Morris-Pratt* menggunakan informasi tersebut untuk membuat pergeseran satu karakter

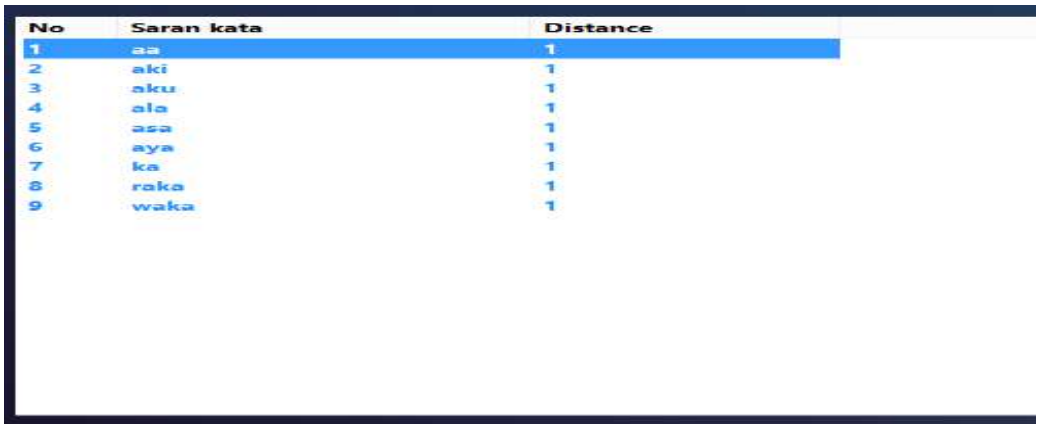
Adapun cara kerja algoritma dalam melakukan pencarian kata adalah sebagai berikut :

1. Algoritma *Knuth-Morris-Pratt* mulai mencocokkan *pattern* pada awal teks.
2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
3. Karakter di pattern dan di teks yang dibandingkan tidak cocok (*mismatch*).
4. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
5. Algoritma kemudian menggeser *pattern* berdasarkan tabel, lalu mengulangi langkah 2 sampai pattern berada di ujung teks .

3. HASIL DAN PEMBAHASAN

3.1 Implementasi

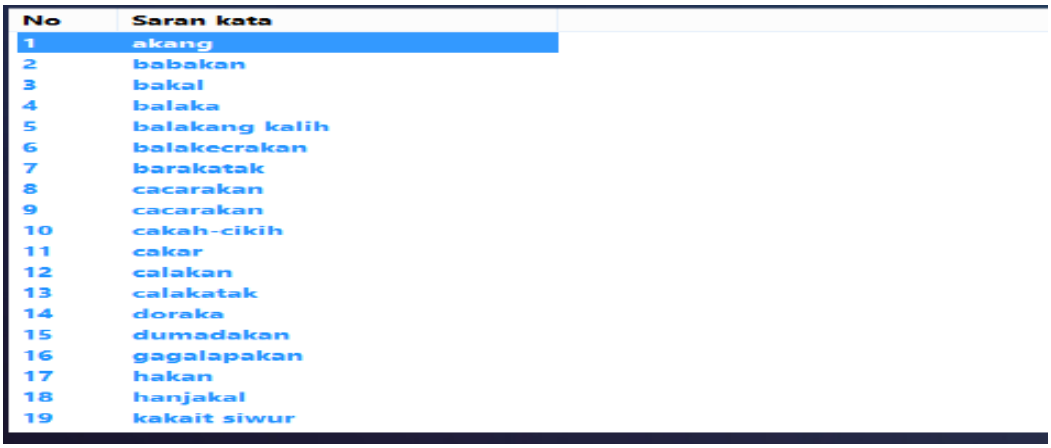
Implementasi adalah kegiatan yang dilakukan untuk menguji data dan menerapkan sistem yang diperoleh dari hasil analisis. Implementasi merupakan salah satu unsur tahapan dari keseluruhan pembangunan sistem komputerisasi, dan unsur yang harus dipertimbangkan dalam pembangunan sistem. Sistem ini dibangun dengan menggunakan bahasa pemrograman VB.NET dengan *IDE* Visual Studio 2013. Gambar 7 menunjukkan halaman pencarian kata algoritma *levenstein distance.*



Gambar 6. Halaman Hasil Pencarian Kata Algoritma *Levenstein Distance*

Halaman hasil proses pencarian kata Algoritma *Knuth-Morris-Prat*t merupakan halaman yang menampilkan hasil pencarian kata *Knuth-Morris-Pratt* Gambar 8. menunjukkan halaman hasil pencarian kata KMP.

E-ISSN : 3024-8752
P-ISSN : 3024-8744



No	Saran kata
1	akang
2	babakan
3	bakal
4	balaka
5	balakang kalih
6	balakecrakan
7	barakatak
8	cacarakan
9	cacarakan
10	cakah-cikih
11	cakar
12	calakan
13	calakatak
14	doraka
15	dumadakan
16	gagalapakan
17	hakan
18	hanjakal
19	kakait siwur

Gambar 7. Halaman Hasil Pencarian Kata *Knuth-Morris-Pratt*

3.2. Pengujian Sistem

Data yang telah di *input* ke dalam sistem berjumlah 6362 kata . Data tersebut merupakan kata yang merujuk pada Kamus Lengkap Bahasa Sunda [2] Pada tabel 3 akan diberikan beberapa data kata pada *database* kamus bahasa sunda . berikut ini adalah tabel data dalam kamus bahasa sunda.

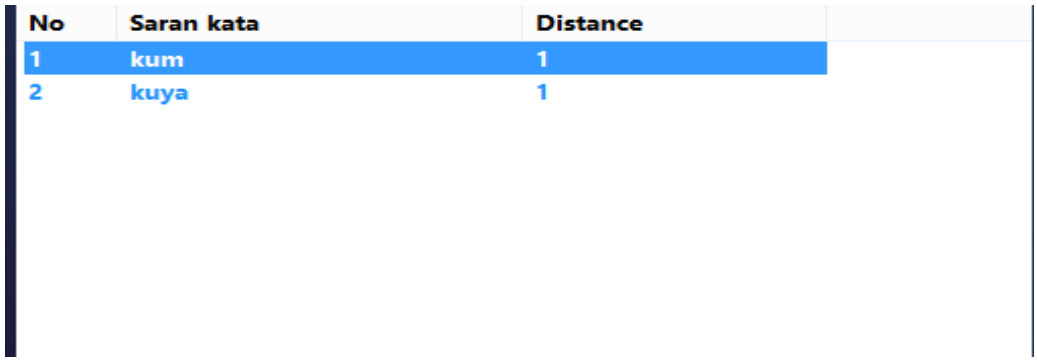
Tabel 3. Sampel Data Kamus Bahasa Sunda

id	sunda	indonesia
2	aa	kakak
3	abah	ayah
4	abdi	saya
5	ameh	supaya
6	aben	adu
7	aber	bepergian
8	ablag	luas
9	ablu	pekerjaannya tidak menentu
10	abong	mentang-mentang
11	abot	berat
12	abotna	beratnya
13	abreg	datang bersamaan waktunya
14	abret	lari sambil melompat
15	abring	berjalan bersama-sama
16	abrug	meronta-ronta
17	abrul	berjalan bersama-sama
18	abur	dilepas
19	abur-aburan	pergi jauh dan berpindah-pindah tempat
20	abus	masuk

Dapat dilihat pada tabel 3 bahwa sampel data yang telah disimpan dalam database adalah sebanyak 20 data kata, dimana data tersebut akan dijadikan sebagai database pada *search engine* dengan algoritma *Levenstein Distance* dan algorima *Knuth-Morris-Pratt.*

Pada algoritma *Levenstein Distance* mencari kata yang mirip dengan indeks kata yang ada pada database contohnya.

E-ISSN : 3024-8752
P-ISSN : 3024-8744



No	Saran kata	Distance
1	kum	1
2	kuya	1

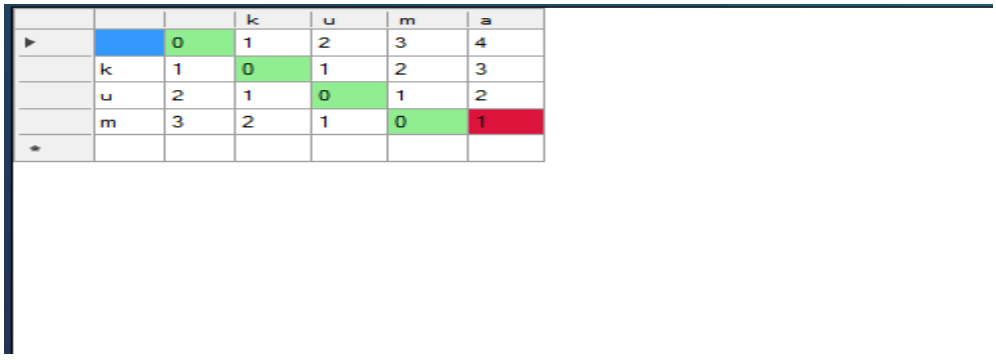
Gambar 8. Pengujian *Leventein Distance*

Berikut hasil pengujian sampel pada gambar 9 pada *fitur word completion* menggunakan algoritma *Levenstein Distance* :

1. Misalnya seorang *user* ingin mengetik kata “kuma”.
2. Lalu saran kata yang diberikan oleh Levenstein Distance adalah “kum”
3. Dimana algoritma *Levenstein Distance* melakukan penghapusan pada huruf “a”.

Jadi algoritma *Levenstein Distance* berfungsi sebagai pengkoreksi kata yang salah dengan cara melakukan penghapusan,penyisipan,substitusi, yang mana apabila string yang cocok diberikan nilai 0 dan string yang berbeda diberikan nilai 1.

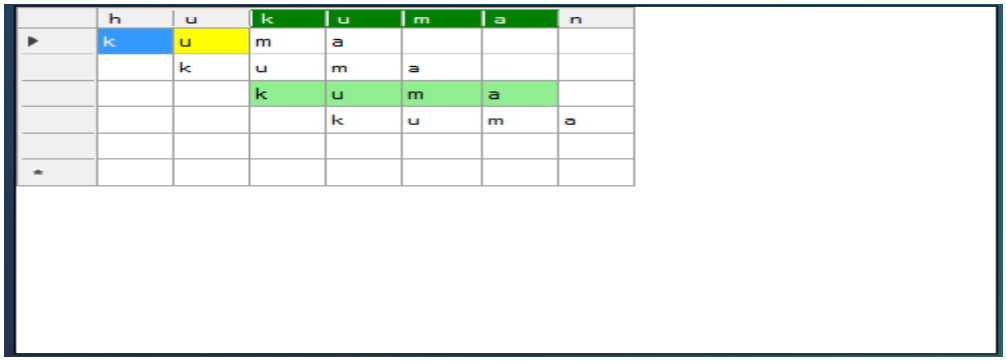
Berikut gambar 10 adalah proses pencarian string yang dilakukan algoritma *Levenstein Distance* :



			k	u	m	a
▶		0	1	2	3	4
	k	1	0	1	2	3
	u	2	1	0	1	2
	m	3	2	1	0	1
*						

Gambar 9. Pergeseran *String Levenstein Distance*

Pada algoritma *Knutt Morris Pratt* menggeser string karakter kesebelah kanan sebanyak 1 karakter sampai string ditemukan.



	h	u	k	u	m	a	n
▶	k	u	m	a			
		k	u	m	a		
			k	u	m	a	
				k	u	m	a
*							

Gambar 10. Pengujian Knutt Morris Pratt Pada Kamus Bahasa Sunda

Berikut hasil pengujian gambar 11. *Word Completion* menggunakan algortima *Knutt Morris Pratt* :

1. Misalnya seorang *user* ingin mengetik kata “kuma”.
2. Lalu saran kata yang diberikan oleh *Knutt Morris Pratt* adalah “hukuman”
3. Dimana algoritma *Knutt Morris Pratt* melakukan pergeseran string yang mengandung kata kuma yaitu”hukuman”
4. Jadi algoritma *Knutt Morris Pratt* berfungsi sebagai pelengkap kata yang dimana setiap kata yang diketikkan hanya beberapa huruf algoritma Knutt Morris Pratt dapat mencari kata yang berhubungan dengan kata yang dicari dengan melakukan pergeseran string kesebelah kanan sebanyak 1 karakter .

4. KESIMPULAN

Implementasi Algoritma *Levenstein Distance* pada *Search Engine* dapat digunakan untuk memperbaiki kesalahan dalam pengetikan dengan cara penghapusan,penyisipan,substitusi. dan mencari kemiripan kata dengan menghitung distance yang ada didatabase dengan kata yang yang dicari. Implementasi Algoritma *Knutt Morris Pratt* pada *Search Engine* dapat digunakan dengan baik karena dapat menemukan kata dengan menseleksi karakter per karakter sampai dinyatakan cocok. Implementasi dalam kedua algoritma tersebut memudahkan pencarian kata tanpa mengetikkan kata seluruhnya. Dengan menggunakan algoritma *Levenstein Distance* dan Algoritma *Knuth Morris Pratt* pencarian kata dapat lebih mudah dan dapat mengkoreksi kesalahan kata dalam pengetikan lalu menseleksi kata sehingga kata dapat lebih mudah untuk dicari.

REFERENSI

[1] J.C, Prasad. & Panicker, K.S.M. 2010. String Searching Algorithm Implementation-Performance Study with Two Cluster Configuration. *International Journal of Computer Science & Communication* 1(2) : 271-275.
[2] Kourie, Justin. Watson, Bruce. & Cleophas, Loek. 2011. On Compile Time Knuth-Morris-Pratt Precomputation. *Proceedings of the Prague Stringology Conference* : 15–29.
[3] Singla, Nimisha. & Garg, Deepak. 2012. String Matching Algorithms and their Applicability in various Applications. *International Journal of Soft Computing and Engineering (IJSCE)* 1(6) : 2231-2307.
[4] Jaiswal,manjid. 2014. Accelerating Enhanced Boyer-Moore String Matching Algorithm on Multicore GPU for Network Security. *International Journal of Computer Applications* 1(9) (0975 – 8887.
[5] SaiKrishna,Vidya . 2012. String Matching and its Applications in Diversified Fields . International Journal of Computer Science Issues (IJCSI) 1(2) : 1694-0814