JARO-WINKLER *DISTANCE* DAN *STEMMING* UNTUK DETEKSI DINI HAMA DAN PENYAKIT PADI

Fairly Okta'mal [1)], Ristu Saptono[2)] Meiyanto Eko Sulistyo[3)]
[1]Informatika, Fakultas MIPA, Universitas Sebelas Maret
Jl. Ir. Sutami No 36 A, Surakarta, 57126
Telp : (0271) 646994, Fax : (0271) 646655
E-mail : perly.oktamal@gmail.com[1)]

Abstrak

Untuk mengurangi terjadinya kesalahan deteksi hama dan penyakit pada tanaman padi, telah dibuat aplikasi dengan menggunakan metode Jaro-Winkler Distance. Untuk menambah akurasi aplikasi dalam pendeteksian hama dan penyakit dalam penelitian ini, selain menggunakan Jaro-Winkler Distance juga menggunakan metode Stemming dengan algoritma Nazief dan Adriani.
Metode stemming digunakan untuk menyederhanakan inputan. Jika inputan tidak sesuai, maka inputan akan diperbaiki dengan mencari kemiripan teks gejala melalui proses pembobotan dengan menggunakan metode Jaro-Winkler Distance. Inputan user yang telah memalui proses stemming dan diperbaiki dengan metode Jaro-Winkler Distance selanjutnya diidentifikasi dengan menggunakan Hamming Distance.
Pada percobaan input pertama didapatkan hasil akurasi sebesar 100%, dan pada percobaan input kedua didapatkan hasil 98%. Pada percobaan output pertama didapatkan nilai akurasi 100%, nilai precision 100% dan nilai recall 100%. Pada percobaan output kedua didapatkan nilai akurasi sebesar 96%, nilai precision 92, 86%, dan nilai recall 96, 15%.

Abstract

To reduce the occurrence of error detection of pests and diseases in rice plant, has created an application using the Jaro-Winkler Distance. To add an application in the detection accuracy of pests and diseases in this study, in addition to using the Jaro-Winkler Distance also using Stemming with a Nazief and Adriani algorithms.
Stemming method is used to simplify input. If the input does not match, then the input will be corrected by searching for text similarity of symptoms through a weighting process using the Jaro-Winkler Distance. User input that has been through a process stemming and improved Jaro-Winkler method Distance subsequently identified using Hamming Distance.
In the first input experiment showed an accuracy of 100%, and the second input experiment showed 98%. In the first experiment obtained output value 100% accuracy, precision value of 100% and 100% recall value. In the second experiment obtained output value amounted to 96% accuracy, precision value 92.86%, and 96.15% recall value.

Keyword: *Jaro-Winkler Distance, Stemming, Hamming Distance*

1. PENDAHULUAN

Tanaman padi yang sehat adalah tanaman padi yang tidak terserang oleh hama dan penyakit [8]. Tetapi pada kenyataannya banyak hama dan penyakit yang menyerang tanaman padi. Saat tanaman padi terserang hama atau penyakit, petani sering mengabaikan karena ketidaktahuannya dan menganggap hal tersebut sudah biasa terjadi pada masa tanaman.

Kesalahan pada penentuan hama atau penyakit yang menyerang tanaman padi berakibat pada kesalahan pada pengendalian hama atau penyakit, sehingga menyebabkan hasil panen berkurang dan bahkan mengakibatkan gagal panen. Untuk mengetahui secara tepat jenis penyakit yang menyerang tanaman padi, diperlukan seorang pakar ahli pertanian sedangkan jumlah pakar pertanian terbatas untuk menangani masalah petani secara bersamaaan. Untuk membantu petani dalam mendeteksi hama padi diperlukan sebuah sistem yang memiliki fungsi seperti seorang pakar, dimana didalam sistem berisi pengetahuan dan dapat mendeteksi jenis hama dan penyakit padi yang diinputkan oleh petani.

Sistem diagnosa penyakit dan hama tanaman padi berbasis web telah dikembangkan oleh beberapa peneliti salah satunya penelitian yang dilakukan oleh Rochmawan [7]. *Input* tekstual deteksi dini hama dan penyakit pada

tanaman padi dengan metode Jaro-Winkler yang telah dilakukan oleh Rochmawan, *input* gejala kurang sederhana, sehingga tingkat akurasi masih rendah [7].

Untuk meningkatkan akurasi pada Jaro-Winkler, pada penelitian ini digunakan metode *stemming* dengan algoritma Nazief dan Andriani. *Stemming* merupakan suatu proses yang terdapat dalam sistem IR yang mentransformasi kata-kata yang terdapat dalam suatu dokumen ke kata-kata akarnya (*root word*) dengan menggunakan aturan-aturan tertentu [1]. Algoritma yang digunakan adalah Nazief dan Andriani karena Algoritma *stemming* untuk bahasa yang satu berbeda dengan algoritma *stemming* untuk bahasa lainnya dan menurut penelitian yang dilakukan oleh Agusta algoritma Nazief dan Andriani memiliki keunggulan dibandingkan algoritma lainnya yaitu memiliki nilai keakuratan (presisi) yang lebih tinggi dan kamus yang digunakan sangat mempengaruhi hasil *stemming* [1]. Jika gejala yang terdapat di dalam database lebih lengkap maka hasil stemming juga akan lebih akurat. Algoritma ini juga memberikan hasil paling baik dan cepat untuk menemukan kata dasar dalam sebuah kalimat [6].

2. DASAR TEORI

2.1 *Preprocesing Input*

Preprocessing adalah tahapan mengubah suatu dokumen kedalam format yang sesuai agar dapat diproses lebih lanjut. Terdapat tiga tahapan proses preprocessing yaitu : *tokenizing, stopword removal, stemming* [9]. Tetapi dalam penelitian ini hanya menggunakan *stemming*.

2.1.1 *Stemming*

Stemming merupakan suatu proses yang terdapat dalam sistem IR *(Information Retrieval)* yang mentransformasi kata-kata yang terdapat dalam suatu dokumen ke kata-kata akarnya (*root word*) dengan menggunakan aturan-aturan tertentu [6]. Algoritma yang dibuat oleh Bobby Nazief dan Mirna Adriani ini memiliki tahap-tahap sebagai berikut [1] :

1. Cari kata yang akan distem dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tesebut adalah *root word*. Maka algoritma berhenti.
2. *Inflection Suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”) dibuang. Jika berupa *particles* (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus *Possesive Pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.
3. Hapus *Derivation Suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a

a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
b. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4

4. Hapus *Derivation Prefix*. Jika pada langkah 3 ada sufiks yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.

a. Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.
b. For i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika root word belum juga ditemukan lakukan langkah 5, jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama algoritma berhenti.

5. Melakukan *Recoding*.
6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

Tipe awalan ditentukan melalui langkah-langkah berikut:

1. Jika awalannya adalah: “di-”, “ke-”, atau “se-” maka tipe awalannya secara berturut-turut adalah “di-”, “ke-”, atau “se-”.
2. Jika awalannya adalah “te-”, “me-”, “be-”, atau “pe-” maka dibutuhkan sebuah proses tambahan untuk menentukan tipe awalannya.
3. Jika dua karakter pertama bukan “di-”, “ke-”, “se-”, “te-”, “be-”, “me-”, atau “pe-” maka berhenti. Jika tipe awalan adalah “none” maka berhenti.

2.1.2 Jaro-Winkler *Distance*

Jaro-Winkler *distance* adalah merupakan varian dari Jaro *distance* metrik yaitu sebuah algoritma untuk mengukur kesamaan antara dua *string*, biasanya algoritma ini digunakan di dalam pendeteksian duplikat [5]. Semakin tinggi Jaro-Winkler *distance* untuk dua string, semakin mirip dengan string tersebut. Jaro-Winkler *distance* terbaik dan cocok untuk digunakan dalam perbandingan *string* singkat seperti nama orang. Skor normalnya seperti (0) menandakan tidak ada kesamaan, dan (1) adalah sama persis.

Algoritma Jaro-Winkler *distance* memiliki kompleksitas waktu *quadratic runtime complexity* yang sangat efektif pada *string* pendek dan dapat bekerja lebih cepat dari algoritma edit *distance*. Dasar dari algoritma ini memiliki tiga bagian:

1. Menghitung panjang *string*.
2. Menemukan jumlah karakter yang sama di dalam dua s*tring*.
3. Menemukan jumlah transposisi.

Pada algoritma Jaro-Winkler digunakan rumus untuk menghitung jarak (dj) antara dua *string* yaitu s1 dan s2 dapat dilihat pada persamaan (1)

$$d_j = \frac{1}{3}\left(\frac{m}{|s1|} + \frac{m}{s2} + \frac{m-t}{m}\right) \quad (1)$$

dimana m adalah jumlah karakter yang sama persis, |s1| adalah panjang *string* 1, |s2| adalah panjang *String* 2, T adalah jumlah transposisi, dan d_j adalah Nilai jarak antara dua buah *string* yang dibandingkan.

Bila mengacu kepada nilai yang akan dihasilkan oleh algoritma Jaro-Winkler maka nilai jarak maksimalnya adalah 1 yang menandakan kesamaan *string* yang dibandingkan mencapai seratus persen atau sama persis. Biasanya s1 digunakan sebagai acuan untuk urutan di dalam mencari transposisi. Yang dimaksud transposisi di sini adalah karakter yang sama dari *string* yang dibandingkan akan tetapi tertukar urutannya.. Sebagai contoh, dalam membandingkan kata CRATE dengan TRACE, bila dilihat seksama maka dapat dikatakan semua karakter yang ada di s1 ada dan sama dengan karakter yang ada di s2 , tetapi dengan urutan yang berbeda. Dengan mengganti C dan T, dapat dilihat perubahan kata CRATE menjadi TRACE.Pertukaran dua elemen *string* inilah adalah contoh nyata dari transposisi yang dijelaskan. Dalam pencocokkan DwAyNE dan DuANE memiliki urutan yang sama D-A-N-E, jadi tidak ada transposisi.

Jaro-Winkler *distance* menggunakan *prefix scale* (p) yang memberikan tingkat penilaian yang lebih, dan *prefix length* (l) yang menyatakan panjang awalan yaitu panjang karakter yang sama dari *string* yang dibandingkan sampai ditemukannya ketidaksamaan. Bila *string* s1 dan s2 yang diperbandingkan, maka Jaro-Winkler *distance*nya (dw) seperti pada persamaan (2)

$$\mathbf{d_w = d_j + (l_p(1 - d_j))} \quad (2)$$

dimana d_w adalah nilai Jaro-Winkler *Distance*, dj adalah Jaro *distance* untuk *strings* s1dan s2, l adalah panjang prefiks umum di awal *string* nilai maksimalnya 4 karakter(panjang karakter ygsama sebelum ditemukan ketidaksamaan *max* 4), dan p adalah konstanta *scaling factor*.

Nilai standar untuk konstanta ini menurut Winkler adalah p = 0,1. Berikut ini adalah contoh pada perhitungan Jaro Winkler *distance*. Jika *string* s1 MARTHA dan s2 MARHTA maka:
m = 6, s1 = 6, s2 = 6
karakter yang tertukar hanyalah T dan H maka t = 1.
Maka nilai Jaro *distance*nya adalah:
dj= 1/3 (6/6 + 6/6 + (6-1)/6) = 0.944
Kemudian bila diperhatikan susunan s1 dan s2 dapat diketahui nilai l = 3, dan dengan nilai konstan p = 0.1. Maka nilai Jaro-Winkler *distance* adalah :
dw = 0.944 + (3 × 0.1 (1 − 0.944)) = 0.961

2.2 Hamming *Distance*

Hamming *distance* merupakan metode untuk mengukur jarak antara dua *string* yang ukurannya sama dengan membandingkan simbol-simbol yang terdapat pada kedua *string* pada posisi yang sama [2]. Hamming *distance* dari dua *string* adalah jumlah simbol dari kedua *string* yang berbeda. Sebagai contoh Hamming *distance* antara *string* ‘toned’ dan ‘roses’ adalah 3. Hamming *Distance* digunakan untuk mengukur jarak antar dua *string binary* misalnya jarak antara 10011101 dengan 10001001 adalah 2.

Haming *distance* digunakan untuk mencari seberapa mirip sebuah vektor terhadap vektor lainnya berdasarkan nilai kedekatannya. Jika nilai kedekatan semakin kecil maka artinya kemiripan kedua vektor semakin besar sebaliknya jika nilai kedekatan semakin besar artinya kemiripan kedua vektor semakin kecil.

3. METODOLOGI

3.1 Studi Literatur dan Pemahaman

Studi literatur dilakukan dengan mengumpulkan bahan referensi tentang Jaro-Winkler *Distance*, *stemming* dengan algoritma Nazief dan Adriani, dan Hamming *Distance* guna memahami bagaimana proses serta cara penerapannya dalam hasil pendeteksian hama dan penyakit.

3.2 Pengumpulan Data

Penelitian ini menggunakan data sekunder berupa data hama dan penyakit, gejala-gejalanya serta cara pencegahan dan penanganannya. Data diambil dari hasil penelitian berupa jurnal, buku yang diterbitkan oleh Puslitbang Tanaman Pangan berserta Balai Pengkajian Teknologi Pertanian dan IRRI (*International Rice Research Institute)*, serta penjelasan pakar hama tumbuhan Fakultas Pertanian Universitas Sebelas Maret Surakarta Ir., Retno Wijayanti, M.Si.

3.3 Pemodelan Data

3.3.1 Proses Identifikasi Input

Input yang dimasukan oleh user akan dicocokan dengan database domain. Jika terdapat kesalahan penulisan aplikasi akan memperbaiki inputan dengan metode Jaro-Winkler *Distance* dengan pembobotan *term* kata, selanjutnya *input*an dicocokan lagi dengan database domain. Setelah didapatkan domain kemudian dicocokan ke database gejala dengan menggunakan *invers* hamming *distance* untuk mendapatkan gejala yang dimaksut oleh user

3.3.2 Proses Penemuan Hama dan Penyakit

Hasil *output* dari proses identifikasi *input*akan digunakan sebagai *feedback* untuk dihitung kemiripannya dengan gejala-gejala yang dimiliki oleh tiap penyakit sehingga akan diperoleh hasil deteksi beberapa penyakit yang mungkin dialami.

3.4 Pengembangan Aplikasi

Tahap implementasi dalam penelitian ini nantinya akan mengggabungkan metode Jaro-Winkler *distance, stemming*, dan metode hamming *distance.* Tetapi dalam penelitian ini yang digunakan adalah *invers* dari Hamming *distance*, yaitu jika benar bernilai (1) dan jika salah berinlai (0). Contoh *invers* Hamming *distance* antara *string* 'toned' dan 'roses' adalah 2. Bahasa pemrograman yang digunakan adalah PHP dan MySQL sebagai *database server*.

3.5 Pengujian dan Analisis Hasil

Pada penelitian ini, pengujian akan dilakukan untuk mengukur ketepatan fitur deteksi. Pengujian akan dilakukan dua kali dengan perhitungan sebagai berikut:

1. Pengujian identifikasi *input*
   Pengujian identifikasi *input* dilakukan untuk mengukur akurasi fitur dalam membuat daftar gejala yang sesuai dengan *input*an *user*. Akurasi akan dihitung menggunakan rumus:
   Akurasi: $\frac{jumlah\ input\ benar}{jumlah\ input\ yang\ dimasukkan} \times 100\%$
2. Pengujian identifikasi *output*
   Pengujian identifikasi *output* dilakukan untuk mengukur akurasi fitur dalam memberi kemungkinan penyakit-penyakit yang sesuai dengan gejala yang sebelumnya sudah di*input*kan. Akurasi akan dihitung menggunakan rumus:
   Akurasi: $\frac{jumlah\ output\ benar}{jumlah\ percobaan\ yang\ dilakukan} \times 100\%$

4. HASIL DAN PEMBAHASAN

4.1. Pemodelan Data

4.1.1. Deteksi *Input* Gejala

Deteksi *input* bertujuan untuk mengindentifikasi *input* gejala oleh *user* dengan data gejala pada aplikasi. *User* meng*input*kan data gejala dengan tanda pemisah (,), (.), (|), (:). Setelah itu *input*an gejala *user* akan dicocokan dengan *database* domain. Jika ada kesalahan penulisan, *input*an terlebih dahulu diproses menggunakan Jaro-Winkler *distance* untuk mencari kata yang dimaksud. Setelah didapatkan *term* kata yang dimaksud kemudian di cocokan dengan gejala yang ada di aplikasi. Berikut adalah contoh tahap deteksi *input*:

Input gejala *user*: daun waran knuign

Tabel 1. *Database* Domain

Id_Domain	Doamin	Domain_stemm
D14	berwarna	warna
D19	daun	daun
D35	kuning	kuning

Dalam *input* diatas kata ‘daun’ akan langsung ditemukan di dalam *database* domain karena tidak ada kesalahan penulisan kata, tetapi kata ‘waran’ dan ’knuign’ harus diproses menggunakan Jaro-Winkler *distance* terlebih dahulu. Berikut adalah perhitungan nilai Jaro-Winkler *distance* ‘waran’ dan ’knuign’
Menghitung nilai Jaro-Winkler *distance* antara ‘waran’ dengan ‘warna’
s1 = waran = 5
s2 = warna = 5
Penentuan transposisi dan karakter yang sama persis

Table 2. Penentuan Transposisi dan Karakter Sama Persis (1)

		0	1	2	3	4
		w	a	R	a	n
0	w	√				
1	a		√			
2	t			√		
3	n					v
4	a				v	

Dari Tabel 2, dapat dilihat bahwa jumlah karakter yang sama persis m = 5, dengan transposisi t = 2/2 = 1, karakter yang bertransposisi adalah a dan n, *prefix length* l = 3
Perhitungan nilai Jaro *distance* dihitung dengan persamaan (1)
$d_j = \frac{1}{3}\left(\frac{5}{5} + \frac{5}{5} + \frac{5-1}{5}\right) = 0.933$
Perhitungan nilai Winkler *distance* dihitung dengan persamaan (2)

d_w = 0.933 + (3 * 0.1(1 - 0.933) = 0.9531

Menghitung nilai Jaro-Winkler distance antara ‘knuign’ dengan ‘kuning’:
s1 = 6
s2 = 6

Dari Tabel 3, dapat dilihat bahwa jumlah karakter yang sama persis m = 6, dengan transposisi t = 4/2 = 2, karakter yang bertransposisi adalah a dan n, *prefix length* l = 1

Table 3. Penentuan Transposisi dan Karakter Sama Persis (2)

		0	1	2	3	4	5
		k	n	U	i	g	n
0	k	√					
1	u			√			
2	n		√				
3	i				v		
4	n						V
5	g					V	

Perhitungan nilai Jaro *distance* dihitung dengan persamaan (1)
$d_j = \frac{1}{3}\left(\frac{6}{6} + \frac{6}{6} + \frac{6-2}{6}\right) = 0.889$
Perhitungan nilai Winkler *distance* dihitung dengan persamaan (2)
d_w = 0.889 + (1 * 0.1(1 - 0.889) = 0.9001

Setelah didapatkan *input*an *user* yang sesuai, kemudian *input*an di lakukan pencocokan dengan *database* gejala oleh sistem dengan Hamming *distance*. Dimana metode yang digunakan adalah *invers* dari hamming *distance*. Contoh perhitungan pencocokan gejala dengan *database*:

Input gejala *user* yang terdeteksi = ‘daun kuning’
Database gejala yang ada = ‘daun berwarna kuning’
Nilai *invers* Hamming *distance*nya adalah 2 dari *term* kata ‘daun’ dan ‘kuning’.

4.1.2 Deteksi *Output*

Gejala yang sudah terdeteksi kemudian akan di cocokan dengan hama atau penyakit yang ada pada *database*. Pencocokan hama atau penyakit juga dilakukan dengan *invers* dari hamming *distance*, yaitu setiap gejala yang terdeteksi terdapat pada hama atau penyakit akan bernilai 1, jika tidak bernilai 0.

4.2 Hasil Pengujian

4.2.1 Pengujian Identifikasi *Input*

Pengujian dilakukan sebanyak dua percobaan, percobaan pertama sebanyak 25 kali dengan kesalahan penulisan, dan percobaan kedua dilakukan sebanyak 25 kali dengan tanpa kesalahan penulisan atau *autocomplete*. Percobaan pertama mendapatkan akurasi identifikasi *input* sebesar 100%. Sedangkan percobaan kedua mendapatkan akurasi *input* sebesar 98%. Berikut daftar tabel pengujan *input*:

Tabel 4. Deteksi *Input* Percobaan Pertama

No Percobaan	No *Input*	*Input User*	Deteksi Sistem	Maksud *User*
1	1	daun gulug	G006	G006
	2	daun wrna putih	G002	G002
2	3	malai mati	G024	G024
	4	busuk gabah	G039	G039
3	5	hampa gbah	G027	G027
	6	Malai berserakn	G038	G038
4	7	daun busuk	G009	G009
	8	malai abuabu	G020	G020
5	9	daun warna kunig	G001	G001
	10	btag warna kunig	G031	G031
15	28	daun wrna kunig	G001	G001
	29	batang wrna kunig	G031	G031

No Percobaan	No *Input*	*Input User*	Deteksi Sistem	Maksud *User*
25	49	mlai abu-abu	G020	G020
	50	padi mdah cabt	G022	G022

Tabel 5. Deteksi *Input* Percobaan Kedua

No Percobaan	No *Input*	*Input User*	Deteksi Sistem	Maksud *User*
1	1	daun bercak putih	G011	G011
	2	daunnya hanya tinggal tulang	G017	G017
2	3	anakan padinya sedikit	G044	G044
	4	daunnya menggulung	G006	G006
	5	daunnya kuning	G001	G001
3	6	malai padinya kecil-kecil	G034	G025
	7	gabah hampa	G027	G027
4	8	tanaman roboh atau rebah	G047	G047
	9	Potongan batang patah berbentuk serong	G032	G032
	10	tikus menyerang dari tengah petak sawah	G048	G048
5	11	gabah hampa	G027	G027
	12	gabah berkerut	G035	G035

	13	gabah berwarna coklat	G036	G036
15	37	gabah hampa	G027	G027
	38	daun terdapat bercak coklat	G012	G012
	39	tangkai malai patah	G023	G023
25	63	bercak coklat kemerahan	G013	G013
	64	tanaman roboh	G047	G047

4.2.2 Pengujian Identifikasi *Output*

Pengujian *output* dilakukan dari pengujian *input* yang dilakukan sebelumnya. Dimana hasil *output* percobaan pertama mendapatkan nilai akurasi sebesar 100%, nilai *precision* 100% dan nilai *recall* 100%. Kemudian pada percobaan kedua mendapatkan nilai akurasi sebesar 96%, nilai *precision* 92, 86% dan nilai *recall* 96, 15%. Berikut daftar table pengujian *output*:

Tabel 6. Deteksi *Output* Percobaan Pertama

No. Percobaan	Deteksi Sistem	Deteksi Pakar
1	PH003	PH003
2	PH014	PH014
3	PH009	PH009
4	PH005	PH005
5	PH001	PH001
15	PH001	PH001
25	PH005	PH005

Tabel 7. Deteksi *Output* Percobaan Kedua

No. Percobaan	Deteksi Sistem	Deteksi Pakar
1	PH011	PH011
2	PH023	PH023
3	PH018	PH006
4	PH004	PH004
5	PH008	PH008
15	PH015	PH015
25	PH013	PH013

4.3 Analisa hasil

Akurasi *input* mempengaruhi akurasi *output*. Dimana pada percobaan pertama menghasilkan akurasi *input* sebesar 100% dan akurasi *output* juga 100%. Tetapi pada percobaan kedua akurasi *input*nya 98% yang menyebabkan akurasi *ouput* menjadi 96%. Kesalahan deteksi *input* pada percobaan kedua karena *input* yang dimasukan *user* kurang sesuai dengan data yang ada pada *database*, yaitu *input* yang dimasukan *user* adalah ‘mali padinya kecil-kecil’ sedangkan yang dimaksud *user* adalah ‘malai pendek’. Gejala yang terdekteksi menjadi ‘pangkal batang bercak kecil kehitaman’. Kesalahan tersebut juga membuat kesalahan identifikasi *output*. Seharusnya penyakit yang terdeteksi adalah ‘kepinding tanah’ karena penyakit ini memiliki gejala ’malai pendek’.

5. SIMPULAN DAN SARAN

5.1 Simpulan

Pengujian identifikasi *input* percobaan pertama menghasilkan akurasi sebesar 100%. Dan aidentifikasi *input* percobaan kedua menghasilkan akurasi sebesar 98% dengan satu kesalahan identifikasi. Hasil identifikasi *output* percobaan pertama adalah 100 % dan percobaan kedua hanya 96% dikarenakan kesalahan hasil identifikasi *input* sebelumnya.

Dalam penelitian ini dapat diketahui bahwa perbedaan *input* yang dimaksut *user* dan *database* masih mempengaruhi hasilidentifikasi *input*.

5.2 Saran

Dalam penelitian ini masih ada kekurangan dalam mengidentifikasi *input* yang dimasukan *user* sehingga berpengaruh dalam mengidentifikasi *output*. Saran untuk memperbaiki aplikasi ini adalah dengan penambahan kosa kata atau sinonim dari kata yang ada dalam *database* untuk memperbaiki akurasi dalam mengindentifikasi *input*.

6. DAFTAR RUJUKAN

[1] Agusta, L. (2009). Perbandingan Algoritma *Stemming* Porter dengan Algoritma Nazief & Adriani Untuk *Stemming* Dokumen Teks Bahasa Indonesia. *Universitas Kristen Satya Wacana.*

[2] Agusta, Y(2008). *Similarity Measure*. http://yudiagusta.wordpress.com/2008/05/13/similarity-measure. Diakses pada tanggal 19 Agustus 2014 11.09 WIB

[3] Kurniawati, A., Puspitodjati, S., & Rahman, S. (2010). Implementasi Algoritma Jaro-Winkler *Distance* untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia.*Universitas Gunadarma*.

[4] Masli, D. L. (2010). Implementation Nazief & Andriani *Stemming Algoritm* to *Calculate Number*. *Faculty of Computer Science Soegijapranata Catholic University*

[5] Rokhmawan, K.W., 2014. *Input* Tekstual Untuk Deteksi Dini Hama dan Penyakit Tanaman Padi Menggunakan Algoritma Jaro Winkler *Distance,* Metode *Association Rule,* dan Metode *Cosine Similarity*. Surakarta: Universitas Sebelas Maret.

[6] Sembiring, A. S. (2013). Sistem Pakar Diagnosa Penyakit dan Hama Tanaman Padi. *Pelita Informatika Budi Darma, Volume III*

[7] Supriyanto, C. & Affandy, 2011. Kombinasi Teknik *Chi Square* Dan *Singular Value Decomposition* Untuk Reduksi Fitur Pada Pengelompokan Dokumen. *Seminar Nasional Teknologi Informasi & Komunikasi Terapan (Semantik)*, (ISBN 979-26-0255-0).

[8] Suyamto. (2007). Masalah Lapang Hama, Penyakit dan Hama pada Padi. *Petunjuk Teknis Badan Litbang Pertanian.*