IMPLEMENTASI ALGORITMA JARO-WINKLER DAN LEVENSTEIN DISTANCE DALAM PENCARIAN DATA PADA DATABASE

Yulianingsih
Program Studi Informatika, Universitas Indraprasta PGRI
Jl. Nangka No.58,Tanjung Barat, Jakarta Selatan
E-Mail: yuliagniya@yahoo.co.id

Abstrak

Data adalah sumber informasi yang harus bernilai atau mengandung makna benar yang dapat digunakan atau diolah hingga memberikan manfaat bagi setiap orang yang membutuhkan. Data dapat ditemukan atau dicari sesuai degan kebutuhan dengan demikian diperlukan pemahaman yang baik mengenai beberapa metode pencarian untuk mendapatkan hasil yang tepat. Dalam penulisan disampaikan hasil dari pengujian komparasi menggunakan algortima Levenshtein Distance dan Jero-Winkler terhadap sejumlah data-data nama yang memiliki tingkat kemiripan yang tinggi. Tujuan dari penelitian ini adalah menentukan satu dari dua metode yang memiliki tingkat validasi tertinggi dalam pencarian data yang memiliki kesamaan tinggi. Pengujian dilakukan secara berulang dan kemudian dilakukan observasi pada tiap tahapan. Temuan hasil dapat digunakan sebagai dasar penentuan penggunaan metode yang tepat dalam pencarian data.

Kata Kunci: Data, Levenshtein Distance, Jaro-Winkler, Kemiripan

Abstract

Data is a source of information that must be valued of contains true meaning which can be used or processed to benefit everyone need. Data can be found or searched in accordance with the needs thus required a good understanding of some search methods to get the right results. In this writing submitted the results of comparative testing using Levenshtein Distance and Jero-Winkler algorithm against a number of human name data which has a high degree of similarity. The purpose of this study is to determine one of two methods that have highest level of validation in search data that has a high similarity.Testing is done repeatedly and then carried out observations at each stage. Finding can be used as a basic for determining the appropriate method of use in data search.

Key Words: *Data, Levenshtein Distance, Jaro-Winkler,Similarity*

Pendahuluan

Data dalam sebuah informasi merupakan sumber pertama yang apabila dikembangkan akan menghasilkan suatu nilai yang bermanfaat bagi orang lain. Saat seorang membutuhkan data dipastikan data tersebut haruslah bernilai benar, dalam pengertian sesuai dengan kebutuhan. Dengan demikian proses pencarian data haruslah sesuai dengan syarat yang dicari. Dan untuk memperoleh kondisi tersebut maka diperlukan pemahaman yang baik mengenai beberapa metode algoritma pencarian.

Diasumsikan bagaimana mengidentifikasi apakah data pelanggan tersebut merupakan pelanggan yang sama dari sekian banyak data pelanggan yang ada. Daftar pelanggan memiliki duplikasi dikarenakan adanya kesalahan ejaan atau kesalahan ketik ketika proses memasukan data. Dengan kata lain pelanggan tersebut seharusnya merupakan pelanggan yang sama. Melalui tulisan ini dilakukan pegujian dan pembuktian bagaimana metode algoritma Levenshtein Distance dan Jaro-Winkler dapat meberikan solusi terhadapat permasalah tersebut. Dengan parameter nama dan tangal lahir yang sama kedua algoritma

digunakan untuk mengukur kesamaan yang dimiliki antara dua string data dan menentukan nilai perkiraan kemiripan data. Dalam penelitian juga ditentukan waktu terbaik yang dimiliki dari kedua algoritma. Diharapkan hasil dari penelitian dapat digunakan sebagai acuan penelitian mengenai algoritma pencarian similarity untuk berbagai kebutuhan lainnya.

Tinjauan Pustaka

Query Database

Query adalah penyelidikan ke database menggunakan pernyataan SELECT. Query digunakan untuk mengekstrak data dari database dalam format yang dapat dibaca sesuai permintaan pengguna. Misalnya, bagaimana mencari data pelanggan pada sebuah tabel yang mempunyai kriteria tertentu. Query ke database untuk informasi pelanggan yang dapat digunakan adalah Query khas yang dapat dilakukan dalam database.

SELECT *, levenshtein(`nama`, "Thomas") + levenshtein(`tgl_lahir`, "1965-05-08") AS score FROM warga ORDER BY score ASC LIMIT 10;

Kalimat diatas merupakan query database dengan parameter pencarian berdasarkan nama mengandung kata “Thomas” dan tanggal lahir “1965-05-08” pada tabel warga dengan mengurutkan hasil pencarian secara ascending dan ditampilkan persepuluh baris.

Matrik String

Matrik string adalah matrik yang digunakan untuk mengukur jarak antara dua string teks untuk mendekati kemiripan atau perbandingan dan pencarian. Beberapa algoritma yang menggunakan konsep dasar matrik string antara lain Levenshtein distance dan Jero-Winkler.

Dalam beberapa bidang ilmu algoritma Levenshtein Distance dan Jaro-Winkler seringkali digunakan untuk proses analisis suatu keadaan antara lain penggunaan dalam deteksi kecurangan, pencocokan RNA atau DNA, idetifikasi plagiarisme, database atau data mining, analisis fingerprint, image dan sebagainya.

Algoritma Levenshtein Distance

Levenshtein Distance atau Edit Distance merupakan metrik string untuk mengukur perbedaan antara dua urutan. Pada algoritma Levenshtein Distance semakin kecil nilai skor yang dimiliki maka kemiripan yang dimiliki semakin tinggi.

Dengan rumus

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1(a_i \neq b_j) \end{cases} \end{cases}$$

Dimana

$1(a_i \neq bj)$ = fungsi indikator

$lev_{a,b}(i,j)$ = jarak pertama karakter dari a dan karakter pertama dari b

Algoritma Jaro-Winkler

Jaro-Winkler merupakan varian dari metrik Jaro Distance biasanya digunakan dibidang keterkaitan rekaman (duplikat) dirancang dan paling sesuai untuk string pendek. Pada Jaro-Winkler untuk dua string semakin tinggi jarak, semakin mirip data yang diperoleh dengan skor 0 sama dengan tidak ada persamaan dan 1 sama persis. Dasar dari algoritma ini

memiliki kriteria antara lain menghitung panjang string, menemukan jumlah karakter yang sama di dalam dua string dan nenemukan jumlah transposisi.

Rumus yang digunakan:

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

Dimana
m = jumlah karakter yang sama persis
s_1 = pangjang string 1
s_2 = pangjang string 2
t = jumlah transposisi (karakter yang sama pada string)
Dan jarak yang dapat dibenarkan adalah jika tidak melebihi

$$\left(max\frac{s1,s2}{s}\right)\leftarrow 1$$

Bila string s_1 dan s_2 yang diperbandingkan maka Jaro-Winkler distancenya (d_w) adalah:
dw = dj + (*l*p(1-dj))
dimana:
dj = Jaro distance untuk strings s1 dan s2
l = panjang prefiks umum di awal string nilai maksimalnya 4 karakter (panjang karakter yg sama sebelum ditemukan ketidaksamaan max 4)
p = konstanta *scaling fact*or. Nilai standar untuk konstanta ini menurut Winkler adalah
p 0,1.

Penelitian lain yang relevan

Menururt Dwitiyastuti dkk (2013) pada sistem pengoreksi ejaan untuk mengatasi terjadinya kesalahan ejaan algoritma Levenshtein Distance dapat diterapkan dengan baik. Pengujian sistem terhadap pencarian kata dalam kamus dilakukan dengan cara memasukkan 14 kata berbeda dengan kemungkinan kata tersebut ejaannya benar atau salah. Parameter keberhasilan pada sistem adalah kesesuaian antara keberadaan kata dalam kamus dengan tampilan kata yang salah pada system.

Menurut Kurniawati (2010) bahwa Algoritma Jero-Winkler dapat diterapkan dengan baik dalam sebuah aplikasi melakukan untuk pendeteksian plagiarisme pada sebuah karya ilmiah berbahasa Indonesia. Pendeteksian dengan urutan kata yang sama aplikasi berhasil menemukan adanya plagiarisme namun ketika urutan kata dibedakan aplikasi tidak mendapati adanya plagiarisme pada sebuah karya ilmiah.

Menurut Oktamal, F dkk(2015)bahwa Implementasi Jaro-Winkler dalam aplikasi mampu mendeteksi hama dan penyakit pada tanaman padi. Dalam penelitian dilakukan kombinasi metode Steaming untuk menyederhanakan inputan kemudian jika inputan tidak sesuai maka inout akan diperbaiki dengan metode Jaro-Winkler selanjutnya diidentifikasi dengan menggunakan Hamming Distance.

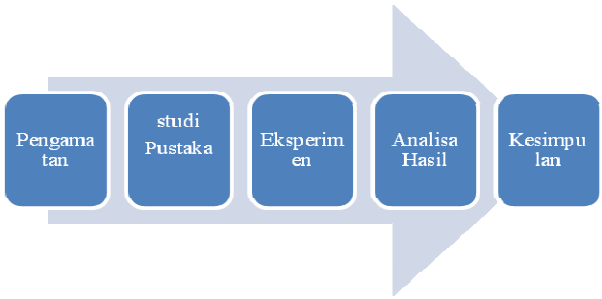
Dita, R.P. (2015) mengatakan bahwa metode approximate string matching menggunakan algoritma levenshtein distance mampu menghitung jarak minimum pengubahan suatu string menjadi string lain dengan optimal. Penelitian juga membahas analisis hasil penerapan algoritma levenshtein distance untuk search suggestion pada aplikasi D-TAB.

Metodologi Penelitian

Data yang digunakan dalam oberservasi adalah 100.000 *record* data menggunakan database MySql (*field* name, tanggal lahir, tempat lahir, no KTP, alamat) dengan parameter pencarian nama dan tgl_lahir. Sumber data pengujian didapat dari *sample Data Spawner Generator*.

Tahapan eksperimen dalam penelitian dilakukan untuk mencari pengaruh perlakuan tertentu terhadap yang lain dalam kondisi yang terkendalikan.

Langkah-langkah penelitian yang digunakan untuk mendapatkan hasil diuraikan berdasarkan urutan proses yang ditunjukan pada gambar 1.



Gambar 1. Langkah-langkah Penelitian

Penelitian dibagi kedalam 5 langkah kegiatan, yaitu: Pengamatan, studi Pustaka, Eksperimen, Analisa Hasil dan Kesimpulan. Berikut uraian dari setiap langkah penelitian.

Langkah ke-1 Pengamatan, merupakan kegiatan yang dilakukan diawal penelitian untuk mendapatkan permasalahan yang dianggap perlu untuk diberikan solusi yang tepat. Berdasarkan pengamatan ditemukan bagaimana metode pencarian data yang tepat yang dapat memberikan hasil terbaik ketika melakukan proses pencarian data tertentu pada suatu kelompok data. Ditentukan dua metode algoritma yang diduga dapat memberikan solusi yaitu metode Levenshtein Distance dan metode Jaro-Winkler.

Langkah ke-2 Studi Pustaka, mencari beberapa kemungkinan solusi ilmiah melalui literatur terpercaya dari sejumlah buku, jurnal dan situs yang dianggap mampu memberikan jawaban yang paling tepat. Selanjutnya melakukan penyusunan *query database. Query* pertama memanggil *stored procedure* yang mengandung algoritma Levenshtein Distance. *Query* kedua memanggil *stored procedure* yang mengandung algoritma Jaro-Winkler.

Langkah 3 Eksperimen
Query pertama algoritma Levenshtein Distance dijalankan pada database kemudian dilakukan pencacatan hasil. Demikian selanjutnya dilakukan perulangan sebanyak lima kali dengan metode yang sama.

Query kedua algoritma Jaro-Winkler dijalankan pada database kemudian dilakukan pencacatan hasil. Demikian selanjutnya dilakukan perulangan sebanyak lima kali dengan metode yang sama. Kedua cara diberlakukan dengan cara yang sama pada kumpulan data yang sama tujuannya untuk mendapatkan kesamaan perlakuan kemudian dilakukan pencatatan dari setiap hasil akhir.

Langkah ke-4 Analisa Hasil, dari beberapa kali pengujian menggunakan dua metode tersebut ditentukan apakah kedua metode memperoleh hasil sesuai dengan yang diharapkan. Selanjutnya dilakukan perbandingan hasil dari kedua metode tersebut.

Langkah ke-5 Kesimpulan, dari sejumlah eksperimen terhadap kedua metode dimbil kesimpulan bahwa metode yang menghasilkan persentasi tertinggi adalah metode terbaik yang mendekati kemiripan dengan nilai parameter dengan waktu pencarian yang optimal.

Hasil dan Pembahasan

Pengujian dengan Algoritma Levenshtein Distance

Menyusun kode algoritma Stored Procedure sebagai berikut

```
BEGIN
DECLARE s1_len, s2_len, i, j, c, c_temp, cost INT;
DECLARE s1_char CHAR;
DECLARE cv0, cv1 VARBINARY(256);
SET s1_len = CHAR_LENGTH(s1), s2_len = CHAR_LENGTH(s2), cv1 = '□', j = 1, i = 1, c
= 0;
IF s1 = s2 THEN
RETURN 0;
ELSEIF s1_len = 0 THEN
RETURN s2_len;
ELSEIF s2_len = 0 THEN
RETURN s1_len;
ELSE
WHILE j <= s2_len DO
SET cv1 = CONCAT(cv1, UNHEX(HEX(j))), j = j + 1;
END WHILE;
WHILE i <= s1_len DO
SET s1_char = SUBSTRING(s1, i, 1), c = i, cv0 = UNHEX(HEX(i)), j = 1;
WHILE j <= s2_len DO
SET c = c + 1;
IF s1_char = SUBSTRING(s2, j, 1) THEN SET cost = 0; ELSE SET cost = 1; END IF;
SET c_temp = CONV(HEX(SUBSTRING(cv1, j, 1)), 16, 10) + cost;
IF c > c_temp THEN SET c = c_temp; END IF;
SET c_temp = CONV(HEX(SUBSTRING(cv1, j+1, 1)), 16, 10) + 1;
IF c > c_temp THEN SET c = c_temp; END IF;
SET cv0 = CONCAT(cv0, UNHEX(HEX(c))), j = j + 1;
END WHILE;
SET cv1 = cv0, i = i + 1;
END WHILE;
END IF;
RETURN c;
END
```

Kemudian dilakukan pemanggilan Stored Procedure dengan Algoritma Levenshtein Distance Dalam Query SELECT *, levenshtein(`nama`, "Thomas") + levenshtein(`tgl_lahir`, "1965-05-08") AS score FROM warga ORDER BY score ASC LIMIT 10;

Hasil Query Dengan Algoritma Levenshtein Distance

Pada Tabel 1ditampilkan hasil query dengan parameter nama yang mengandung kata Thomas dan tgl_lahir pada 1965-05-08 dan ditampilkan dalam setiap sepuluh baris *record*.

Tabel 1. Hasil Query Levenshtein Distance

id	nama	tgl_lahir	tpt_lahir	no-ktp	alamat	skore
80938	Thomas York	1965-05-08	Fresno	660-61-5290	15025 North Holy See (Vatican City State) Ave.	7
25470	Kato Roman	1965-08-08	Marquette	154-70-4907	31666 East Guyana Ct.	9
33854	Ava Mays	1961-05-09	St. George	902-56-8003	28970 Gardena Ln.	9
88430	Seth Mays	1996-09-09	Minnetonka	161-53-4121	48947 East Oman Ave.	9
23404	Laith Dale	1965-05-06	San Rafael	777-85-4453	35123 North Turkey Ave	10
36623	Uta Hogan	2015-05-03	Niaga Falls	395-67-3695	48758 South Bell Way	10
44278	Oren Melton	1965-05-08	Vincennes	641-08-6153	50234 North Virgin Islands, British Way	10
65359	Suki Thomas	1971-09-08	Seattle	548-63-3467	22346 West American Samoa Way	10
67617	Thomas Holt	1963-05-08	Hollister	628-27-0989	59106 South Monrovia Blvd.	10
81741	Lara Thomas	1955-02-02	Pasco	095-02-5895	45442 West Gambia Ave.	10

Berdasarkan tabel 1 ditemukan bahwa urutan nama teratas terdiri dari nama yang mengandung “Thomas” dan kombinasi kata “Thomas” pada urutan selanjutnya, pada pengujian algoritma Levenshtein Distance semakin kecil nilai skor yang dihasilkan maka kemiripan yang dimiliki semakin tinggi.

Pengujian dengan Algoritma Jaro-Winkler

Menyusun kode algoritma Stored Procedure sebagai berikut

```
BEGIN
#finestra:= search window, curString:= scanning cursor for the original string, curSub:=
scanning cursor for the compared string
declare finestra, curString, curSub, maxSub, trasposizioni, prefixlen, maxPrefix int;
declare char1, char2 char(1);
declare common1, common2, old1, old2 varchar(255);
declare trovato boolean;
declare returnValue, jaro float;
set maxPrefix=6; #from the original jaro - winkler algorithm
set common1="";
set common2="";
set finestra=(length(in1)+length(in2)-abs(length(in1)-length(in2))) DIV 4
+ ((length(in1)+length(in2)-abs(length(in1)-length(in2)))/2) mod 2;
set old1=in1;
set old2=in2;
#calculating common letters vectors
set curString=1;
while curString<=length(in1) and (curString<=(length(in2)+finestra)) do
set curSub=curstring-finestra;
if (curSub)<1 then
set curSub=1;
end if;
set maxSub=curstring+finestra;
if (maxSub)>length(in2) then
set maxSub=length(in2);
end if;
```

```
set trovato = false;
while curSub<=maxSub and trovato=false do
if substr(in1,curString,1)=substr(in2,curSub,1) then
set common1 = concat(common1,substr(in1,curString,1));
set in2 = concat(substr(in2,1,curSub-1),concat("0",substr(in2,curSub+1,length(in2)-
curSub+1)));
set trovato=true;
end if;
set curSub=curSub+1;
end while;
set curString=curString+1;
end while;
#back to the original string
set in2=old2;
set curString=1;
while curString<=length(in2) and (curString<=(length(in1)+finestra)) do
set curSub=curstring-finestra;
if (curSub)<1 then
set curSub=1;
end if;
set maxSub=curstring+finestra;
if (maxSub)>length(in1) then
set maxSub=length(in1);
end if;
set trovato = false;
while curSub<=maxSub and trovato=false do
if substr(in2,curString,1)=substr(in1,curSub,1) then
set common2 = concat(common2,substr(in2,curString,1));
set in1 = concat(substr(in1,1,curSub-1),concat("0",substr(in1,curSub+1,length(in1)-
curSub+1)));
set trovato=true;
end if;
set curSub=curSub+1;
end while;
set curString=curString+1;
end while;
#back to the original string
set in1=old1;
#calculating jaro metric
if length(common1)<>length(common2)
then set jaro=0;
elseif length(common1)=0 or length(common2)=0
then set jaro=0;
else
#calcolo la distanza di winkler
#passo 1: calcolo le trasposizioni
set trasposizioni=0;
set curString=1;
while curString<=length(common1) do
if(substr(common1,curString,1)<>substr(common2,curString,1)) then
set trasposizioni=trasposizioni+1;
```

```
end if;
set curString=curString+1;
end while;
set jaro=
(
length(common1)/length(in1)+
length(common2)/length(in2)+
(length(common1)-trasposizioni/2)/length(common1)
)/3;
end if; #end if for jaro metric
#calculating common prefix for winkler metric
set prefixlen=0;
while (substring(in1,prefixlen+1,1)=substring(in2,prefixlen+1,1)) and (prefixlen<6) do
set prefixlen= prefixlen+1;
end while;
#calculate jaro-winkler metric
return jaro+(prefixlen*0.1*(1-jaro));
END
```

Pemanggilan Stored Procedure dengan Algoritma Jaro-Winkler Dalam Query
SELECT *, jaro_winkler(`nama`, "Thomas") + jaro_winkler(`tgl_lahir`, "1965-05-08") AS score FROM warga ORDER BY score DESC LIMIT 10;

Hasil Query Dengan Algoritma Jaro-Winkler

Tabel 2 merupakan hasil query dengan parameter nama “Thomas” dan tgl_lahir “1965-05-08” ditampilkan dalam setiap sepuluh baris record hasil.

Tabel 2. Hasil Query Jaro-Winkler

id	nama	tgl_lahir	tpt_lahir	no-ktp	alamat	skore
80938	Thomas York	1965-05-08	Fresno	660-61-5290	15025 North Holy See (Vatican City State) Ave.	‘1.9393937
67617	Thomas Holt	1963-05-08	Hollister	628-27-0989	59106 South Monrovia Blvd	‘1.8538383841
81897	Thomas Gentry	1953-04-08	Waterbury	116-65-0946	75534 West Northern Mariana Islands Way	‘1.8215384483
23447	Thomas Bright	1956-01-20	Effingham	081-96-0885	33624 East Reunion ST.	‘1.78820514
38763	Thomas Odom	1977-03-08	St. Marys	539-42-9043	47968 Athens Ln.	‘1.77939397
45996	Thomas Wolfe	1951-04-07	Bradford	475-63-1377	72008 Des Moines ST.	‘1.77333337
74059	Thomas Beach	1953-07-01	Homer	623-49-7497	22732 Columbus Ave.	‘1.77333337
77695	Thomas Moody	1972-04-08	Ansonia	479-60-9006	80279 South Mauritius Ave.	‘1.77333337
99199	Thomas Curtis	1986-08-04	Jacksonville	246-30-2472	96528 Noth Cape Verde Blvd	‘1.7715384
16802	Thomas Kinney	1982-04-08	Oakland	365-15-1886	82353 West Guinea-bissau Ln.	‘1.7682051

Berdasarkan tabel 2 ditemukan bahwa terdapat sejumlah nama yang mengandung “Thomas” ditempatkan pada urutan teratas demikian selanjutnya. Pada pengujian algoritma Jaro-Winkler semakin besar nilai skor yang dimiliki maka kemiripan yang dimiliki semakin tinggi.

Kemudian untuk memastikan dan memperkuat hasil temuan maka dilakukan pegujian kembali sebanyak 5 iterasi terhadap 5 nama dan tgl_lahir yang berbeda dan menghasilkan sejumlah temuan sebagaimana yang ada pada tabel 3.

Tabel 3. Hasil Perbandingan Waktu

nama	tgl_lahir	waktu	
		Levenshtein	Jaro-Winkler
Thomas	1965-05-08	155.7721 sec	116.3347 sec
Robert	1963-05-04	153.3558 sec	76.2600 sec
William	1965-05-08	164.7141 sec	85.8108 sec
Dorothea	1995-02-07	176.8801 sec	89.9443 sec
Margareth	1997-08-06	190.9766 sec	99.7793 sec

Pada tabel 3 dapat diamati bahwa dari lima kali pengujian pada masing-masing algoritma ditemukan hasil capaian waktu yang berbeda untuk lima parameter berbeda. Algoritma Jari-Winkler memperoleh waktu lebih cepat dibandingkan dengan algoritma Levenshtein Distance.

Simpulan dan Saran

Simpulan

Dari hasil pengamatan selama melakukan pengujian dapat diambil kesimpulan sebagai berikut :

1. Pada algoritma Levenshtein Distance semakin kecil nilai skor yang dimiliki maka kemiripan semakin tinggi.
2. Pada algoritma Jaro-Winkler semakin besar nilai skor yang dimiliki maka kemiripan semakin tinggi.
3. Algoritma Jaro-Winkler memiliki kompleksitas waktu *quadratic runtime complexity* yang sangat efektif pada string pendek dan dapat memperoleh hasil lebih cepat dari algoritma Levenshtein Distance.
4. MetodeJaro-Winklermudah untuk diimplementasikan dan efektif dalam hasil yang dicapai.
5. Penelitian yang dilakukan dapat digunakan sebagai dasar menentukan penggunaan metode pencarian data untuk beberapa jenis kegiatan lainnya yang serupa.

Saran

Hasil penelitian dapat digunakan sebagai dasar menentukan pemilihan algoritma pencarian terbaik. Diharapkan adanya penelitian lanjutan untuk mengembangkan hasil sehingga algoritma Jaro-Winkler dapat diimplementasikan pada objek yang berbeda.

Daftar Pustaka

Dita, R.P. (2015). Implementasi Algoritma Levenshtein Distance Untuk Search Suggestion Pada Aplikasi Pengelolaan Infformasi Kerusakan Mobil (D-TAB). *Jurnal Ilmiah Universitas Bakrie*, 3 (3).

DuBois, P.(2014). *MySQLCookBook*. (3rd Edition). Publisher : O'REILLY

Dwitiyastuti,R.N, Muttaqin, A dan Aswin,M.(2013).Pengoreksi Kesalahan Ejaan Bahasa Indonesia Menggunakan Metode Levenshtein Distance.Malang: Universitas Brawijya.1(2)

http://www.informit.com/articles/article.aspx?p=29661

http://www.sqlservercentral.com/articles/Fuzzy+Match/65702/

Kurniawati, A. Puspitodjati, S, Rahman,S.(2014).Implementasi Algoritma Jaro-Winkler Distance untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia.Depok: Universitas Gunadarma.

Oktamal, F. Saptono, R dan Sulistyo, M.E.(2015) Jaro-Winkler Distance dan Steaming untuk Deteksi Dini Hama Dan Penyakit Padi. Seminar Nasional Sistem Informasi Indonesia (SESINDO),Departemen Sistem Informasi Institut Teknologi Sepuluh November.

Sugiyono.(2009).Metode Penelitian Kuantitatif Kualitatif dan R&D.Bandung : Alfabeta