Analisis dan Implementasi Pencocokan String Berdasarkan Kemiripan Pengucapan (*Phonetic String Matching*) Menggunakan Algoritma *Metaphone* Dalam Pencarian Ayat Al-Qur'an

Tegar Graha Adiwiguna[1], Ir. Moch. Arif Bijaksana, MTech[2], Shaufiah S.T, M.T[3]

[1]Prodi S1 Teknik Informatika, Fakultas Informatika, Universitas Telkom
[2]Fakultas Informatika, Universitas Telkom
[3]Fakultas Ilmu Terapan, Universitas Telkom
[1]tegargraha@gmail.com, [2]shaufiah@gmail.com, [3]arifbijaksana@gmail.com

Abstrak

Telah ditemukan penelitian baru yang membuat Al-Qur'an dalam versi digital. Akan tetapi, pada umumnya perangkat lunak yang telah ada hanya menggunakan teknik *Exact String Matching* untuk melakukan pencarian informasi (ayat). Dan jika pengguna perangkat lunak ini salah dalam penulisan inputan maka perangkat lunak tidak akan memberikan solusi dari apa yang diinginkan oleh pengguna. Oleh karena itu, tujuan penelitian ini adalah membangun sistem pencarian dengan teknik fonetik atau pencocokan kata berdasarkan pengucapan (*Phonetic String Matching*) yang dapat digunakan untuk mengatasi permasalah tersebut.

Dengan menggunakan Algoritma *Metaphone* dan *Dice Similarity*, sistem pencarian ayat Al-Qur'an ini akan melakukan pencocokan string berdasarkan pengucapan dengan nilai *precision* sebesar 54% dan nilai *recall* sebesar 100%. Juga korelasi yang didapatkan sebesar 82%.

1. Pendahuluan

Al-Qur'an adalah Firman Allah yang diturunkan kepada Nabi Muhammad SAW yang menerangkan hukum hukum, aqidah/i'tiqadiyah, nasehat dan lain lain[2]. Al-Qur'an merupakan kitab suci agama islam yang menjadi pedoman hidup semua umat beragama islam. Al-Qur'an ditulis menggunakan bahasa arab. Al-Qur'an terdiri atas 114 surat, 6.236 ayat, dan 77.845 kata[7]. Dengan jumlah surat, ayat, dan kata yang begitu banyak, pencarian surat, ayat atau kata yang dilakukan secara manual cukup menyulitkan. Seiring dengan perkembangan jaman, kemajuan teknologi dapat dimanfaatkan untuk mengatasi masalah tersebut. Dengan teknik komputasi pencarian surat, ayat, atau kata pada Al-Qur'an dapat dilakukan dengan mudah.

Pada saat ini sudah banyak perangkat lunak yang dibuat untuk memudahkan kita mempelajari Al-Qur'an, terutama yang dapat diunduh pada smartphone saat ini. Sehingga semua pengguna smartphone dapat dengan mudah mempelajari Al-Qur'an. Namun pada saat ini, perangkat lunak yang telah ada hanya dapat menampilkan daftar dari surat-surat yang ada di dalam Al-Qur'an saja, sehingga jika ingin mencari berdasarkan ayat atau kata yang terkandung dalam Al-Qur'an sangat sulit. Oleh karena itu untuk mengatasi kesulitan pengguna dalam pencarian ayat tersebut, perlu dikembangkan sistem pencarian yang dapat digunakan berdasarkan ayat atau kata-kata yang terkandung dalam Al-Qur'an. Namun masih ada kendala yang dihadapi ketika pengguna ingin melakukan pencarian, misalnya salah dalam pengetikan saat memasukan kata yang dimaksud dalam pencarian, oleh karena itu dapat digunakan pencarian kata berdasarkan kemiripan dalam pengucapan, sehingga dapat meminimalisir kesalahan.

Dengan pencocokan string berdasarkan kemiripan dalam pengucapan (*Phonetic String Matching)*, pencarian teks ayat dalam Al-Qur'an dapat dilakukan dengan kata kunci berupa pengucapan kata yang diwakili dengan penulisan huruf latin. Selain itu, pencarian bersifat toleran terhadap perbedaan penulisan pengucapan yang mungkin terjadi, seperti Allah, Aloh, Allaah, dan sebagainya.

Algoritma yang digunakan dalam metode *Phonetic String Matching* ini yaitu algoritma *Metaphone*. Dan untuk mengukur tingkat kemiripan dari kata yang dicari dengan menggunakan algoritma *Dice Similarity*. Dice *Similarity* adalah suatu cara untuk melakukan perhitungan tinggi atau rendahnya kemiripan *(Similarity)* antara dua dokumen, teks, atau kata dengan .

Dari permasalahan yang telah dipaparkan di atas, pada penelitian ini dibangunlah sistem pencarian ayat Al-Qur'an menggunakan metode *Phonetic String Matching* dengan algoritma *Metaphone*, serta pengukuran tingkat kemiripan dengan menggunakan algoritma *Dice Similarity*.

2. Landasan Teori dan Perancangan

2.1. Pedoman Alih Aksara Arab ke Latin

Beraneka ragam lafal dan tulisan yang digunakan dalam transliterasi dari bahasa Arab walaupun memiliki makna yang sama. Pedoman alih aksara Arab ke latin ini disusun agar dapat memahami perbedaan perbedaan tersebut.

2.2. Pencocokan String

Dalam ilmu komputer string adalah deretan karakter. Walaupun sering juga dianggap sebagai data abstrak yang menyimpan sekuens nilai data, atau biasanya berupa bytes yang mana merupakan elemen yang digunakansebagai pembentuk karakter sesuai dengan encoding karakter yang disepakati seperti ASCII, ataupun EBCDIC.. String dapat berupa kata, frase,atau kalimat. Pencocokan String adalah sebuah permasalahan dalam menemukan pola susunan karakter string di dalam string lain atau bagian dari isi teks[5]. Pencocokan string merupakan bagian utama dalam proses pencarian string. Dalam pencocokan string dibedakan menjadi dua, yaitu :

1. Exact string matching, merupakan pencocokan string secara tepat dengan susunan karakter dalam string yang dicocokkan memiliki jumlah maupun urutan karakter dalam string yang sama[1]. Contoh : kata step akan menunjukkan kecocokan hanya dengan kata step.

2. Inexact string matching atau Fuzzy string matching, merupakan pencocokan string secara samar, maksudnya pencocokan string dimana string yang dicocokkan memiliki kemiripan dimana keduanya memiliki susunan karakter yang berbeda (mungkin jumlah atau urutannya) tetapi string-string tersebut memiliki kemiripan baik kemiripan tekstual/penulisan (approximate string matching) atau kemiripan ucapan (Phonetic String Matching).

2.3. Phonetic String Matching

Phonetic String Matching merupakan suatu teknik pencocokan string yang membandingkan suatu string dengan string yang lain berdasarkan kode fonetis masing-masing[5].

2.4. Algoritma Metaphone

Algoritma Metaphone merupakan salah satu algoritma yang diterapkan pada phonetic string matching. Algoritma ini dipublikasikan oleh Lawrence Philips dalam sebuah artikel berjudul "Hanging on the Metaphone" dalam jurnal "Computer Language" vol. 7 n. 12, Desember 1990, pp. 39-43. Algoritma Metaphone menghasilkan kode fonetis yang panjang karakternya berbeda yang disesuaikan dengan panjang string masukan[1]. Algoritma Metaphone merupakan algoritma yang melakukan penangan khusus terhadap setiap fonem (satuan bunyi bahasa) dalam kata. Algoritma Metaphone dapat diterapkan pada berbagai bahasa di dunia dalam pencarian data tertentu. Algoritma Metaphone bertujuan untuk mencari kata-kata yang memiliki kemiripan bunyi baik dalam bahasa Indonesia sendiri maupun pendekatan dalam bahasa Inggris, seperti kata-kata ilmiah dalam bahasa Indonesia masih banyak yang disadur dari bahasa Inggris, seperti contoh dibawah ini : Mikro, Micro, Makro, Macro. Pada contoh kata tersebut apabila mencari data tentang mikro, maka akan keluar data mikro dan micro. Hal itu terjadi karena adanya kemiripan dalam pengucapan kata mikro. Cara kerja Metaphone yaitu kalimat diubah menjadi sebuah kode terlebih dahulu, kemudian apabila ada kemiripan dalam pengucapan maka data yang akan dicari akan ditampilkan[6].

2.4.1. Aturan Metaphone

Aturan-aturan dasar dari Algoritma Metaphone ini adalah sebagai berikut[6]:

1. Menghilangkan semua karakter diluar alfabet.
2. Alfabet yang digunakan hanya 16 suara konsonan yaitu:
   B, F, H, J, K, L, M, N, P, R, S, T, W, X, Y, (kosong) O adalah simbul untuk suara yang dihasilkan oleh "th"
3. Menghilangkan semua huruf vokal A, I, U, E, O
4. Melakukan konversi kata menjadi kode fenotis dengan melakukan pengecekkan setiap huruf yang sesuai dengan aturan bahasa tertentu.

Pada algoritma Metaphone, pemberian kode fonetis memperhatikan juga interaksi antara konsonan dan vokal dalam kata serta kelompok konsonan bukan hanya sebuah konsonan.

2.5. Hashing

Hashing adalah suatu cara untuk mentransformasi sebuah string menjadi suatu nilai yang unik dengan panjang tertentu (fixed-length) yang berfungsi sebagai penanda string tersebut. Fungsi untuk menghasilkan nilai ini disebut fungsi hash, sedangkan nilai yang dihasilkan disebut nilai hash. Contoh sederhana hashing terdapat pada Tabel 2.1.

Tabel 2.1 Contoh sederhana *hashing*

Kata	Nilai hash
qul	12578
ahad	108470
lam	11879

Contoh diatas adalah pengunaan hashing dalam pencarian pada database. Apabila tidak di-hash, pencarian akan dilakukan karakter per karakter pada nama-nama yang panjangnya bervariasi dan ada 26 kemungkinan pada setiap karakter. Namun pencarian akan menjadi lebih cepat setelah di-hash karena hanya akan membandingkan hanya dengan 10 kemungkinan setiap angka. [4].

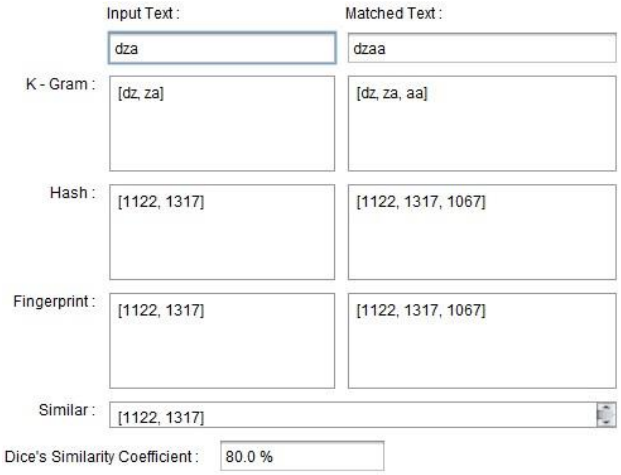
2.6. Dice Similarity Coeficients

Untuk menghitung nilai Similarity dari dokumen fingerprint yang didapat maka digunakan Dice Similarity Coeficients dengan cara menghitung nilai dari jumlah K-Gram yang digunakan pada kedua kata yang diuji. Dimana K-Gram adalah rangkaian karakter atau kata yang diekstrak dari sebuah teks atau kata-kata menjadi dua atau tiga kata secara terurut, sebagai contoh K-Gram yang telah diset bernilai dua dari kata ‘rumah’ maka akan menjadi, ‘ru’, ‘um’, ‘ma’, ‘ah’. Dan jika K-Gram yag telah diset bernilai tiga maka akan menjadi, ‘rum’, ‘uma’, ‘mah’. Sedangkan dokumen fingerprint adalah nilai hash yang unik yang didapat dari jumlah nilai hash pada K-Gram yang berbeda. Nilai Similarity tersebut dapat dihitung dengan menggunakan:

$$\int = \frac{2 \square}{\square\square} \ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

Dimana S merupakan nilai Similarity, dan C merupakan jumlah fingerprint yang sama dari dua buah teks yang dibandingkan, sedangkan A, B merupakan jumlah fingerprint dari masing-masing teks yang dibandingkan[8].

Contoh perhitungan Similarity dengan K-Gram yang diset bernilai 2 dapat dilihat pada Gambar 2.1.



Gambar 2.1 Contoh perhitungan *Similarity*

Dari perhitungan diatas maka didapatkan nilai *Similarity* antara kedua teks tersebut sebesar 80%.

2.7. ASCII

Kode Standar Amerika untuk Pertukaran Informasi atau ASCII (American Standard Code for Information Interchange) merupakan suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|". ASCII selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks.

Nilai hash yang akan dicari dengan fungsi hash dalam algoritma Dice Similarity Ceoficient merupakan representasi dari nilai ASCII (American Standar Code for Information Interchange) yang menempatkan angka numerik pada karakter, angka, tanda baca dan karakterkarakter lainnya. ASCII menyediakan 256 kode yang dibagi kedalam dua himpunan standar dan diperluas yang masing-masing terdiri dari 128 karakter. Himpunan ini merepresentasikan total kombinasi dari 7 atau 8 bit, yang kemudian menjadi angka dari bit dalam 1 byte. ASCII standar menggunakan 7 bit untuk tiap kode dan menghasilkan 128 kode karakter dari 0 sampai 127 (heksadesimal 00H sampai 7FH). Himpunan ASCII yang diperluas menggunakan 8 bit untuk tiap kode dan menghasilkan 128 kode tambahan dari 128 sampai 255 (heksadesimal 80H sampai FFH).

Algoritma Dice Similarity Coeficient melakukan perhitungan nilai hash dengan memperlakukan setiap substring sebagai sebuah angka dengan basis tertentu, di mana basis yang digunakan pada umumnya merupakan bilangan prima yang besar. Misalnya, jika substring yang ingin dicari adalah “dia” dan basis yang digunakan adalah 101, nilai hash yang dihasilkan adalah 100 x 102 + 105 x 101 + 97 x 100 = 11147 (nilai ASCII dari ‘d’ adalah 100,

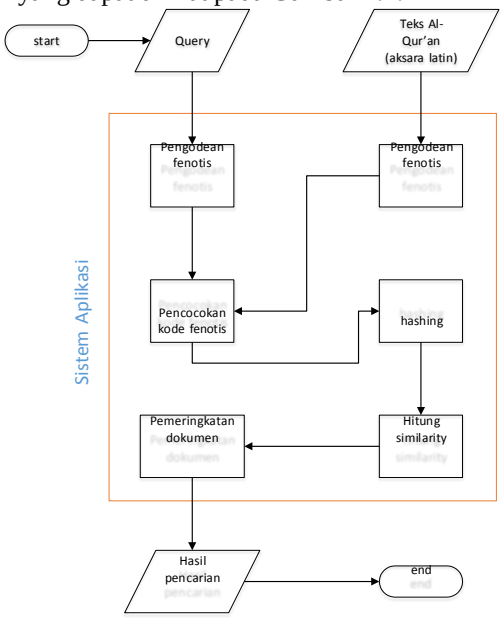
'i' adalah 105, dan nilai ASCII dari 'a' adalah 97). Berikut karakter ASCII yang digunakan pada penelitian ini yang telah disajikan pada tabel 2.2 :

Tabel 2.2 Tabel Karakter ASCII

Kara-kter	nilai unicode (heksades-imal)	nilai ANSI ASCII (desimal)	Keterangan
-	002D	45	Karakter hyphen (strip)
'	0027	39	Karakter Apostrof
(	0028	40	Tanda kurung buka
)	0029	41	Tanda kurung tutup
a	0061	97	Huruf latin a kecil
b	0062	98	Huruf latin b kecil
c	0063	99	Huruf latin c kecil
d	0064	100	Huruf latin d kecil
e	0065	101	Huruf latin e kecil
f	0066	102	Huruf latin f kecil
g	0067	103	Huruf latin g kecil
h	0068	104	Huruf latin h kecil
i	0069	105	Huruf latin i kecil
j	006A	106	Huruf latin j kecil
k	006B	107	Huruf latin k kecil
l	006C	108	Huruf latin l kecil
m	006D	109	Huruf latin m kecil
n	006E	110	Huruf latin n kecil
o	006F	111	Huruf latin o kecil
p	0070	112	Huruf latin p kecil
q	0071	113	Huruf latin q kecil
r	0072	114	Huruf latin r kecil
s	0073	115	Huruf latin s kecil
t	0074	116	Huruf latin t kecil
u	0075	117	Huruf latin u kecil
v	0076	118	Huruf latin v kecil
w	0077	119	Huruf latin w kecil
x	0078	120	Huruf latin x kecil
y	0079	121	Huruf latin y kecil
z	007A	122	Huruf latin z kecil

2.8. Perancangan Sistem

Dalam penelitian ini akan dibangun sebuah sistem yang dapat mengimplementasikan metode pencocokan string berdasarkan pengucapan dengan algoritma Metaphone yang digabungkan dengan algoritma Rabin Karp yang berfungsi sebagai indexing berdasarkan kemiripan antar kata pada pencarian ayat Al-Qur'an. Pencocokan string yang telah diubah menjadi kode fenotis telah didapat melalui proses preprocessing data, sehingga dihasilkan kode fenotis bagi setiap data yang ada dalam database/corpus dan query. Selanjutnya pengukuran tingkat kemiripan menggunakan algoritma Dice Similarity Coeficient, dan indexing berdasarkan besarnya persentase kemiripan antar kata tersebut. Adapun gambaran umum sistem yang dapat dilihat pada Gambar 2.2.



Gambar 2.2 Gambaran umum sistem

2.8.1 Perancangan algoritma Metaphone

Tahapan proses algoritma Metaphone dalam merubah string pada dataset menjadi sebuah kode fenotis adalah sebagai berikut :

1. Menghilangkan semua karakter diluar alphabet
2. Menghilangkan semua huruf vokal kecuali huruf vokal terletak pada awal kata.
3. Melakukan konversi kata menjadi kode fenotis dengan melakukan pengecekkan setiap huruf yang sesuai dengan aturan yang Metaphone yang telah ditentukan pada tabel 2.3.

Tabel 2.3 Aturan Metaphone

AWAL	AKHIR	CONTOH
H	dihapus ketika hruf depannya s, g, t, d,z	yubshiruun, ghaibas-samaawaati, azhlama,
		Faazallahumaasy-syaithaanu, ba'dhukum
	H	haqqa, wa-innahaa
T	dihapus ketika huruf setelahnya s	tsummaat-takhadztumul, Tsumma
	D, ketika ada di akhir kata	Ahad
	T	tasykuruun, aatainaa
Y	dihapus ketika huruf depannya s	tasykuruun, syi-atum, tasytaruu
	I, ketika huruf sebelumnya z	aziizil, mawaaziinuh
	Y	biaayaatii, yazhunnuuna, yauman
K	H, ketika huruf setelahnya h	takhadztumul, biittikhaadzikumul
	k	baari-ikum, anfusakum
D	D	ardhi, faaridhun, ardha
	Z, ketika huruf setelahnya Z	kadz-dzabtum, 'alaal-ladziina
ng (ikhfa)	n	Minhum
V, P	F	Fiima
Q	K	qiyaamati, Walaqad, waqaffa-inaa
J	Z	Dzalika
B	B	ghaibas, ihbithuu, ba'dhukum
G	G	naghfir, bighadhabin
	dihapus ketika huruf depannya n	minhum (minghum)
L	L	minallahi, bighairil
M	M	bimaa, aamanuu
N	N	Aamanuu
	m, ketika huruf setelahnya b (iqlab)	min ba'dii
R	R	wannashaara, ajruhum
S	S	miitsaaqakum, khaasiriin
W	W	Walaqad, wamaa
F	F	khalfahaa, faaridhun
K	K	bikrun, dzalika
Z	Z	Dzalika

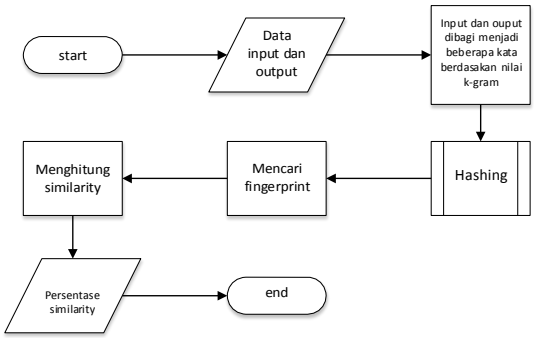
Dari aturan tersebut secara sederhana dapat dibuat pemadanan ke kode fenotis seperti pada tabel 2.4 berikut ini :

Tabel 2.4 Padanan ke kode fenotis

aksara latin	padanan
sh, ts, sy	s
kh	h
zh, dz	z
dh	d
th	t
gh	g
ng(ikhfa)	n
f,v,p	f
q,k	k
j,z,zy	z

2.8.2 Perancangan algoritma Dice Similarity Coeficient

Dalam penerapan algoritma Dice Similarity Coeficient terdapat beberapa proses yaitu proses membagi satu kata menjadi beberapa kata sesuai dengan k-gram yang telah ditentukan, mencari nilai hash (hashing), mencari fingerprint dari nilai hash yang telah ditemukan, menghitung nilai dice Similarity, dan mendapatkan besarnya persentase kemiripan antar teks. Gambaran alur proses yang terjadi pada algoritma Dice Similarity Coeficient telah dijelaskan pada Gambar 2.3.



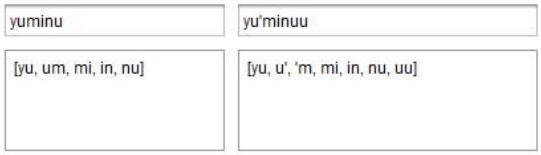
Gambar 3.3 Flowchart Algoritma Dice *Similarity* Coeficient

2.6.1 Ilustrasi Keseluruhan Sistem

Pada tahap ini akan diberikan penjelasan lebih detail mengenai semua proses yang terjadi dari awal aplikasi dijalankan hingga menghasilkan output yang diharapkan.

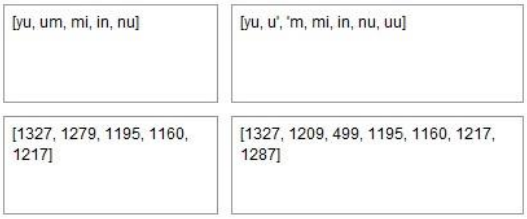
1. Aplikasi di run dan pada tampilan awal akan dilakukan proses input dataset dengan menekan tombol browse file, dan masukan dataset yang telah disediakan sebelumnya.
2. Dataset akan langsung diencode perkata dengan aturan metaphone yang telah dibuat dan disimpan pada array list.
3. User memasukkan input atau query berupa kata (potongan ayat Al-Qur'an), misalnya "yuminu".
4. Dan ketika menekan tombol cari maka kata yuminu akan langsung diencode menjadi kode fenotis yakni "ymn". Dan akan dilakukan pencocokan dengan dataset yang sebelumnya sudah diencode.
5. Aplikasi akan mencari kata yang memiliki kode fenotis yang sama yaitu "ymn". Dan kata yang memiliki kode fenotis tersebut akan disimpan dalam array yang isinya adalah nomor surat, nomor ayat, dan teks ayat tersebut.
6. Setelah pencarian selesai, akan dilakukan perankingan output dengan menggunakan algoritma Dice Similarity berdasarkan nilai similarity secara descending. Proses pencarian nilai similarity adalah sebagai berikut :
   Misal : Input teks : yuminu

   Match teks : yu'minuu
   a. Kata input dan output akan dibagi menjadi beberapa kata sesuai dengan nilai K-Gram yang telah diset sebelumnya (default K-Gram = 2). Maka kata tersebut akan menjadi seperti Gambar 3.6 berikut :



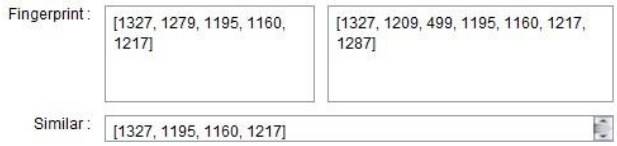
Gambar 3.7 Proses pembagian berdasarkan nilai K-Gram

   b. Dari kata yang telah dibagi menjadi gram-gram tersebut akan dicari nilai hashnya dengan fungsi hash yang merupakan representasi dari nilai ASCII. Proses untuk mendapatkan nilai ASCII adalah sebagai berikut :
   "yu" , nilai hash yang dihasilkan adalah 121 x 101 + 117 x 100 = 1327 (nilai ASCII dari 'y' adalah 121, 'u' adalah 117). Begitu seterusnya hingga didapatkan seperti pada Gambar 3.7



Gambar 3.8 Proses mendapatkan nilai hash

   c. Setelah mendapatkan nilai hash, akan dicari fingerprint dari kedua teks tersebut.
   d. setelah mendapatkan fingerprint, maka akan dicari nilai hash yang sama (similar) diantara kedua teks tersebut. Dan didapatkan hasilnya seperti pada Gambar 3.8



Gambar 3.9 Fingerprint dan similar

   e. Dari data yang telah didapatkan maka akan dilakukan perhitungan persentase similarity dengan menggunakan persamaan 2.1. Untuk lebih jelasnya perhitungannya adalah sebagai berikut :

$$S = \frac{2C}{A+B}$$

Dik :C=4, A=5, B=7

$$\text{jawab : } S = \frac{2*4}{5+7} = 0{,}667$$

   maka didapatkan nilai similarity antara kedua teks sebesar 66,7%.

7. Proses 6 akan terus dilakukan berulang pada semua output yang didapatkan. Setelah semua nilai

persentase setiap output didapatkan maka akan diurutkan secara descending.

8. Hasil pencarian akan ditampilkan pada aplikasi.
9. Selesai.

2.9. Dataset / Korpus

Al-Quran transliterasi diperoleh melalui hasil penulisan responden yang dibandingkan dengan Al-Quran transliterasi yang sudah ada dan aturan alih aksara arab ke latin. Dan pada penelitian ini dataset yang digunakan berupa file teks dengan format .txt. Dalam file tersebut memiliki beberapa atribut yaitu nomor surat, nomor ayat, dan teks ayat dalam aksara latin yang terdiri dari juz satu, dua, dan 30. Tahap perancangan database digunakan untuk merancang tabel yang dibuat, sehingga input dan output aplikasi sesuai dengan yang dibutuhkan. Terdapat sifat pada dataset yaitu seperti penambahan tanda baca (') sebelum huruf "A" yang menandakan dibaca seperti huruf "ع" pada huruf aksara arab.

Tabel 2.2 Contoh ayat transliterasi latin pada *corpus*

مهنْ	Innahum
لاْاومتكتنْ	Wataktuumul
نولءاستيْمْ ع	'Amma yatasaa-aluun.

3. Pengujian dan Analisis

3.1 Tujuan Pengujian

Pengujian yang akan dilakukan adalah pengujian akurasi dengan mencari tingkat *precision* dan *recall*. Pengujian akurasi dilakukan dengan tujuan untuk mengetahui performansi dari algoritma *Metaphone* dalam menemukan kata yang memilki kemiripan pengucapan dalam bacaan Al-Qur'an yang dilatinkan. Pengujian dilakukan berdasakan input yang dilakukan oleh user dan output yang dikeluarkan oleh sistem. Aspek yang akan dievaluasi pada pengujian ini, yaitu akurasi dengan mencari besarnya nilai *precision* dan *recall*, serta mencari nilai korelasi. Selain itu, juga akan dilakukan pengujian aplikasi dengan memberikan berbagai variasi query yang akan didapatkan dari 15 orang responden.

3.1 Hasil Pengujian dan Analisis Skenario Pertama

Pada skenario ini dilakukan pengujian dengan mengukur besarnya nilai precision dan korelasi pada aplikasi dengan melihat besarnya persentase kemiripan antara input dengan output. Pengujian ini bertujuan untuk mencari nilai akurasi precision dan nilai korelasi antara sistem dengan pengurutan secara manual. Berikut alur pengujian yang telah dilakukan untuk mencari nilai precision:

1. Uji query yang telah ditentukan pada aplikasi sebagai inputan.
2. Aplikasi akan melakukan pengindeksan output berdasarkan nilai persentase secara *descending.*
3. Akan diberikan threshold, dimana jika persentase > 50% maka kata tersebut dianggap benar, dan sebaliknya, bila persentase ≤ 50% maka kata tersebut dianggap salah.
4. Didapatkan output yang telah dianggap benar dan salah berdasarkan threshold yang telah ditentukan.
5. Mencari nilai akurasi

Setelah nilai precision didapatkan maka akan dicari nilai korelasi dari hasil pengujian sebelumnya. Berikut alur pengujian yang telah dilakukan untuk mencari nilai korelasi:

1. Peneliti akan melakukan pengecekan secara manual dengan cara output dari aplikasi diurutkan secara random.
2. Output yang sudah diurutkan secara random diberikan kepada responden untuk diurutkan kembali berdasarkan tingkat *similarity* sesuai dengan penilaian responden itu sendiri.
3. Hasilnya akan dibandingkan antara pengurutan yang dihasilkan oleh sistem dan *manual user* (responden).
4. Jika banyak kesamaan antara keduanya, maka aplikasi dianggap memiliki nilai korelasi yang cukup tinggi.

Hasil pengujian *precision* dan korelasi yang telah dilakukan dipaparkan pada Tabel 3.1 dan Tabel 3.2.

Tabel 3.1 Hasil *Precision*

no	query	nilai precision
1	famaa	0.85
2	siro	0.17
3	inalloha	1
4	khola	0
5	tsuma	0.8
6	musa	0.82
7	yuminu	0.6
8	angfusahum	0.8
9	badukum	0.5
10	saiin	0
11	Jalika	1

12	Waija	0
13	Dunillohi	0.5
Precision rata-rata		0.54

Tabel 3.2 Hasil Korelasi

input	nilai korelasi
famaa	1
siro	0.4
khola	1
tsuma	1
waija	-0.4
musa	1
yuminu	1
angfusahum	1
badukum	1
saiin	1
dunillohi	1

Dari pengujian tersebut didapatkan nilai presisi rata-rata sebesar 0,54. Dari hasil pengujian terdapat beberapa kata yang seharusnya benar namun dianggap salah, karena persentase yang dihasilkan antara kata input dan output kurang dari 50%. Hal itu disebabkan karena banyaknya perbedaan nilai hash antara kedua kata tersebut setelah dibagi bagi berdasarkan nilai k-gramnya. Kata yang dianggap tidak sama biasanya terdapat pada kata yang mengandung padanan fenotis yang terdiri dari dua konsonan dan tanda bacaan seperti dz, sy, kh, strip (-), kutip (‘), dsb.

Dan hasil pengujian nilai korelasi didapatkan nilai rata rata sebesar 0,82. Dimana nilai korelasi sangat baik jika bernilai 1, dan sangat buruk jika bernilai -1. Maka dari pengujian ini, nilai korelasi antara sistem dan user dinilai baik dengan nilai korelasi rata-rata 0,82.

3.2 Hasil Pengujian dan Analisis Skenario Kedua

Pada pengujian skenario kedua telah didapatkan query yang dituliskan oleh 15 orang secara random. Query didapatkan dengan cara menuliskan apa yang telah dibacakan oleh penulis dan kata yang dimaksud telah ditentukan, sehingga didapatkan beberapa variasi penulisan query yang berbeda-beda namun maknanya sama. Berikut hasil pengujian skenario kedua yang disajikan pada Tabel 3.2:

Tabel 3.2 hasil skenario kedua

Kata	query	keterangan
Fihii فِيهِ	viihi	benar
	fi..hi	benar
	fiihi	benar
	vvihi	benar
	fihi	benar
matsaluhum مَثَلُهُمْ	mafthaluhum	salah
	matsaluhum	benar
	mashaluhum	benar
	mathaluhum	salah
	matsaluuhum	benar
	masaluhum	benar
	mataluhum	salah
innahum إِنَّهُمْ	innahum	benar
	in..nahum	benar
	in'nahum	salah
	inahum	benar
	innahum	benar
	inna'hum	benar
	innahum(a)	benar
yakhthafu يَخْطَفُ	yakhtafu	benar
	yakhtofu	benar
	yakhthofu	benar
	yahtofu	benar
	yakhthafu	benar
	yahtafu	benar
haitsu حَيْثُ	haishtu	salah
	haitsu	benar
	khaetu	salah
	haistsu	benar
	haitzu	benar
	khaitsu	benar
	haisu	benar
wataktuumul وَتَكْتُمُوا الْ	wataqtummul	benar
	wataq'tummul	benar
	wataqtumul	benar
	wataktumul	benar
	wataktu..mul	Benar

yu'linuun يُعْلِنُونَ	yulinun	benar
	yuklinun	salah
	yu'linuun	benar
	yu'linun	benar
	yuqlinuun	salah
anfusi أَنْفُسِ	angfushi	benar
	an'fusi	benar
	angfutsi	benar
	angfusi	benar
	angfusyi benar	anfusi
	benar angfussi	benar
	angfusii benar	badin
	benar	bakdin
ba'dhin بَعْضِ	salah ba'din	benar
	ba'dinn	benar
	ba'dzim	salah
	ba'dhin benar	ba'dim
	salah yasya'u	benar
	yasyaa'u	benar
	yasyaau	salah
yasyaa-u يَشَاءُ	yasyau	salah
	yaasau salah	yashaa'u
	benar	yasa..u
	salah yasaau	salah
	adabun salah	azabun
	benar adzaabun	benar
	adzabun	benar
	azzabun	benar
adzaabun عَذَابٌ	adzabun	benar
	a'dzaabun	benar
	adza'bun	benar

Dari Tabel 4.3 diatas terdapat beberapa query yang salah atau kata yang dimaksud tidak dapat ditemukan. Dari hasil analisis, hal tersebut terjadi karena kode fenotis yang didapat dari query tidak ada yang sesuai dengan dataset. Dan jika dibaca kembali, query tersebut memiliki bunyi yang berbeda dan tidak sesuai dengan kata yang telah ditentukan.

Untuk query ”yasyaau” yang memiliki bunyi dan huruf yang sama namun dianggap salah, itu disebabkan query tidak menggunakan tanda apostrof (‘) sehingga pengkodean fenotisnya berbeda. Aturan itu dibuat karena untuk membedakan cara membaca pada Al-Qur’an latin. Contohnya untuk membedakan antara huruf alif (ا) dan ain (ع) pada kata “ma’akum” dan “maakum”. Dari kedua kata tersebut memiliki huruf yang sama namun memiliki makna yang berbeda sama halnya dengan kata “yasyau” dan “yasya’u”.

3.3 Hasil Pengujian dan Analisis Skenario Ketiga

Pada skenario ini dilakukan pengujian recall. Dalam pengujian ini ada beberapa tahapan yang dilakukan. Berikut tahap-tahap pengujian yang telah dilakukan :

1. Peneliti menentukan query yang akan diuji.
2. Peneliti melakukan penelusuran secara manual dengan membaca Al-quran yang sudah dilatinkan untuk mencari semua kemungkinan output query yang dimaksud.
3. Query diinputkan kedalam sistem.
4. Setelah itu akan dibandingkan antara output yang dihasilkan oleh sistem dan penelusuran yang dilakukan secara manual, jika jumlah kata dan letak ayat tersebut sama, maka nilai recall yang diujikan tinggi.
5. Nilai recall didapatkan dengan cara jumlah output yang dihasilkan dari sistem aplikasi dibagi dengan jumlah kata yang didapatkan dari penelusuran secara manual.

Dan hasil pengujian ini dapat dilihat pada tabel 3.3 dan tabel 3.4.

Query : wamin
Kode Fenotis : wmn

Tabel 3.3 Tabel hasil pengujian skenario ketiga (user)

surat ke-	ayat	matching
QS.99	8	Waman
QS.113	3	Wamin
QS.113	4	Wamin
QS.113	5	Wamin

Tabel 3.4 Tabel hasil pengujian skenario ketiga (sistem)

no	surat ke-	ayat	Matching	*Similarity*

1	QS.113	3	Wamin	100
2	QS.113	4	Wamin	100
3	QS.113	5	Wamin	100
4	QS.99	8	Waman	50

Dari hasil pengujian keduanya didapatkan jumlah dan hasil pencarian yang sama, sehingga didapatkan nilai recall sebesar 100%.

4. Kesimpulan

Berdasarkan hasil pengujian yang diperoleh dan analisis yang telah dilakukan dapat diberikan kesimpulan dari tujuan yang ada sebagai berikut:

1. Membangun sistem pencarian ayat Al-Quran berbasis kemiripan fonetis. Telah dihasilkan aplikasi sistem pencarian ayat Al-Quran berbasis kemiripan fonetis untuk Al-Quran transliterasi latin yang sesuai pelafalan orang Indonesia dengan metode n-gram dan pengodean fonetis.
2. Mengembangkan metode pengodean fonetis untuk teks Al-Quran yang sesuai untuk pembicara bahasa Indonesia. Telah dikembangkan metode pengodean fonetis untuk teks Al-Quran transliterasi latin yang sesuai pembicara orang Indonesia.
3. Mengukur kinerja sistem pencarian ayat Al-Quran berbasis kemiripan fonetis. Didapatkan kinerja dari sistem yang sudah dibuat dan yang paling baik adalah menggunakan indeks Al-Quran yang menggunakan vokal dengan nilai presisi 0.746 dan menggabungkan metode pemeringkatan jumlah dan posisi.

5. Daftar Pustaka

[1] M. Syaroni and R. Munir, "Pencocokan String Matching Berdasarkan Kemiripan Ucapan (*Phonetic String Matching*) dalam Bahasa Inggriss," in Seminar Nasional Aplikasi Teknologi Informasi, Yogyakarta, 2005.

[2] http://jaibnajhan.blogspot.com/2012/12/pengertian-alquran-kitab-suci-ummat.html // alquran

[3] E. V. Haryanto,"Rancang Bangun *Prototype* Mesin Pencari String Menggunakan Metode *Fuzzy String Matching,"in* Konferensi Nasional Sistem dan Informatika 2011, Bali, 2011.

[4] http://elib.unikom.ac.id/files/disk1/596/jbptunikompp-gdl-pujipraram-29787-10-unikom_p-3.pdf // rabin karp

[5] T. A. Purnamasari,"Membangun Aplikasi Pencocokan String Berdasarkan Penulisan dan Pengucapan," Yogyakarta, 2012

[6] http://akudikampusbiru.wordpress.com/2009/01/11/algoritma-*Metaphone*-anti*Metaphone*/ //*Metaphone*

[7] http://id.wikipedia.org/wiki/Surah

[8] Surahman Ade Mirza, "Perancangan Sistem Penentuan *Similarity* Kode Program Pada Bahasa C dan Pascal dengan Menggunakan Algoritma Rabin-Karp," in Program Studi Teknik Informatika Fakultas Teknik Universitas Tanjungpura, Pontianak, 2013.

[9] http://id.wikipedia.org/wiki/Al-Qur'an

[10]https://id.wikipedia.org/wiki/wikipedia:pedoman_alih_aksara_arab_ke_latin

[11] Istiadi Muhammad Abrar, "Sistem Pencarian Ayat Al-Quran Berbasis Fenotis," in departemen ilmu komputer fakultas matematika dan ilmu pengetahuan alam Institut Pertanian Bogor, Bogor, 2012.