Pattern Matching dalam Aplikasi SimSimi

Diah Fauziah - 13512049
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
diah.fauziah@students.itb.ac.id

***Abstract*—Kesepian dan butuh teman bicara yang harus selalu ada terkadang menjadi permasalahan bagi beberapa orang. Manusia pada umumnya memiliki keterbatasan fisik untuk selalu ada 24 jam dengan tujuan sekedar menjadi teman bicara dan *sharing* satu sama lain. Namun seiring dengan perkembangan teknologi, sekarang manusia tidak hanya bisa berbicara atau *chatting* dengan manusia, tetapi juga dapat berbicara dengan robot pintar. Robot pintar ini memiliki kemampuan menjawab obrolan dari manusia dengan berbagai jawaban yang unik. Salah satu robot pintar yang terkenal saat ini yaitu SimiSimi. SimiSimi merupakan aplikasi yang menerima pertanyaan dari pengguna dan menjawab pertanyaan pengguna sehingga terjadi percakapan. Aplikasi ini menggunakan penerapan *string matching* sebagai dasar dari aplikasi tersebut**

***Index Terms*—SimiSimi, *pattern matching*.**

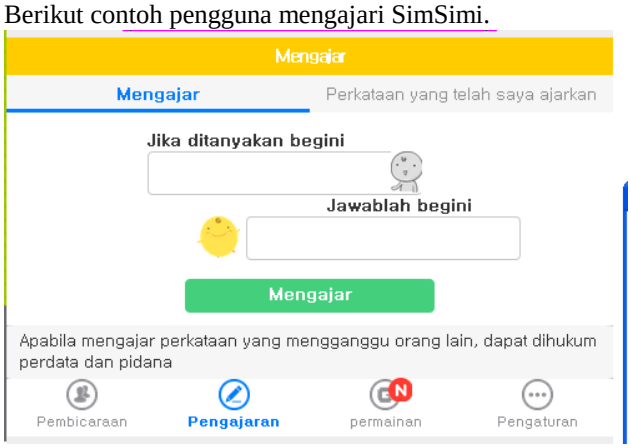
I. PENDAHULUAN

Teman bercerita merupakan hal yang sangat penting dalam kehidupan banyak orang. Berbagai masalah yang dialami membuat banyak orang membutuhkan teman untuk mencurahkan isi hati yang mereka rasakan. Banyak orang yang memilih untukbercerita kepada sahabat,orang tua,kakak,teman, dan orang-orang terdekat lainnya. Namun sebagai manusia biasa, kita tidak bisa selalu bercerita kepada sahabat,teman,dan orang-orang terdekat tersebut. Sebagai manusia biasa kita memiliki kesibukan masing-masing sehingga kita tidak bisa selalu ada 24 jam ada untuk orang-orang terdekat. Hal ini membuat banyak orang butuh teman yang selalu ada, baik itu saat sedih dan senang, sehingga walaupun tidak bersama orang terdekat namun bisa sedikit menghibur.

Kemajuan teknologi dibidang informasi telah menghasilkan berbagai aplikasi yang memenuhi berbagai kebutuhan manusia, termasuk dalam kebutuhan untuk bercerita mencurahkan isi hati setiap saat. Aplikasi yang dapat memenuhi kebutuhan ini salah satunya yaitu *SimiSimi*. SimiSimi bisa didapatkan dengan mudah dengan mengunduh dari ponsel maupun PC. SimiSimi cukup populer terutama bagi kalangan remaja. Tidak jarang SimiSimi memberikan jawaban yang lucu, dan aneh, sehingga membuat penggunanya ketagihan. SimSimi juga dapat diajari pertanyaan berikut jawabannya oleh pengguna. Lucu dan simpelnya SimSimi menjadikan seolah-olah pengguna benar-benar merasa sedang bercerita dengan teman satu sama lain.

SimSimi adalah aplikasi *chatting* buatan Korea yang mengimplementasikan *artificial intilijent* (kecerdasan buatan) dan *pattern matching* (pencocokan string). Pattern matching digunakan saat mencocokkan pertanyaan yang diberikan pengguna dengan pertanyaan yang ada pada *database* SimSimi. Jika tidak ada, maka SimSimi akan mengeluarkan respon "I have no respon. Please teach me". Selanjutnya, antara satu percakapan (terdiri dari satu pertanyaan dan satu jawaban) dengan percakapan lain diimplementasikan kecerdasan buatan. Berikut contoh percakapan antara pengguna dengan SimSimi.

Berikut contoh pengguna mengajari SimSimi.



Gambar 2. Mengajarkan SimSimi

Saat SimSimi diajari sesuatu, pertanyaan dan jawaban yang pengguna ajarkan akan disimpan di database sistem.

II. PATTERN MATCHING

Pattern matching adalah suatu cara untuk mencari atau mencocokkan pola tertentu dalam suatu hal (misalnya teks dan gambar). Definisinya yaitu sebagai berikut.

Diberikan:

1. *T*: teks (*text*) atau gambar, yang panjang/jumlahnya *n* karakter/gambar
2. *P: pattern*, yaitu *text* atau *image* dengan panjang/jumlahnya *m* karakter/gambar (asumsi *m* <<< *n*) yang akan dicari di dalam teks/gambar.

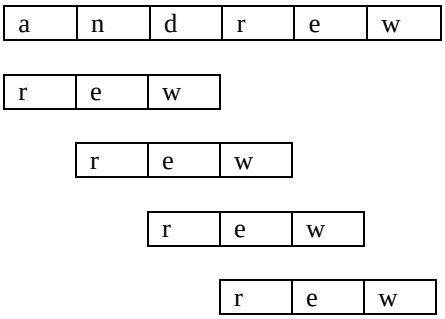
Pattern matching akan mencari teks atau gambar tertentu dan lokasinya di dalam kumpulan teks atau gambar. Pada makalah ini terkait dengan aplikasi SimSimi akan dibahas lebih jauh tentang Pattern matching dengan menggunakan teks / string matching.

Ada banyak algoritma untuk melakukan pencarian string di dalam suatu teks, beberapa diantaranya yaitu Algoritma Brute Force, Algoritma Knuth Morris Patt, dan Algoritma Booyer Moore. Berikut pembahasan ketiga algoritma :

A. AlgoritmaBrute Force

Algoritma ini memeriksa satu per satu posisi pada teks untuk memeriksa kesamaan karakter per karakter antara string pada pattern dan string dengan teks. berikut langkah-langahnya :

1. Periksa karakter awal pattern dan karakter teks
2. Jika sama :
   lanjutkan periksa ke karakter selanjutnya di pattern dan selanjutnya di teks sampai semua karakter di periksa kesamaannya.
   Jika saat pencocokan ditemukan ketidaksamaan, lanjutkan ke langkah 3.
   jika semua pattern telah diperiksa, lanjutkan ke langkah 4.
3. Geser semua pattern 1 langkah ke kanan, lakukan langkah 1
4. Jika ditemukan kesamaan sampai semua pattern selesai diperiksa, pattern ditemukan dan pattern berada di dalam teks. Pencarian selesai.



Pemeriksaan satu per satu karakter dilakukan sampai semua *pattern* ditemukan atau semua karakter di teks diperiksa.

Algoritma ini bagus untuk kasus dimana setiap karakter *pattern P* tidak pernah sama dengan karakter teks T yang dicocokkan atau apabila karakter dapat ditemukan dalam sekali pemeriksaan di setiap karakter pada pattern. Jumlah perbandingan maksimal karakter adalah maksimal karakter pada teks yang diperiksa.

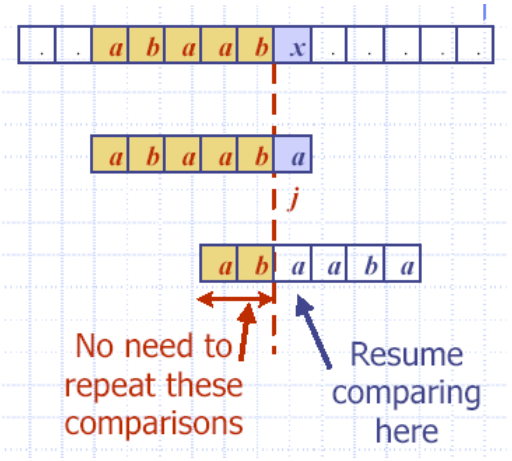
B. Algoritma Knuth Morris Patt

Algoritma ini memeriksa teks dengan pattern dari kiri ke kanan seperti algoritma brute force. Perbedaan signifikan antara algoritma brute force dan Knuth Morris Patt terletak pada kemungkinan pergeseran yang dilakukan. Algoritma Knuth Morris Patt dapat bergeser dan berpindah lebih dari satu langkah sedangkan algoritma brute force hanya bergeser paling banyak satu langkah saja dalam sekali pemeriksaan.

Langkah-langkah algoritma Knuth Morris Patt yaitu sebagai berikut :

Menentukan banyaknya pergeseran akan dilakukan seperti berikut.

1. Awalnya diperiksa dari awal kesamaan karakter pada pattern dengan karakter pada teks.
2. Jika sama lanjut diperiksa selanjutnya dari kiri ke kanan,
   Jika tidak masuk ke langkah 3.
3. Jika pada karakter ke-j pattern tidak cocok dengan string, cari jumlah maksimal pergeseran dari pola agar mengefisiekan pemeriksaan. cara mencari jumlah maksimalnya adalah sebagai berikut.
   Carilah prefix terpanjang dari pola Pattern ke-i hingga karaketer ke- j-1 yang juga merupakan suffiks dari pola Pattern ke-i hingga karakter ke- j-1.



Gambar 3. Simulasi Algoritma KMP

Untuk mencari prefix dan suffiks dari contoh pada gambar diatas, temukan prefix dari abaab (karakter pertama hingga ke j-1) yang juga merupakan suffiks dari abaab. Didapatkan

prefiksnya yaitu ab. “ab” memiliki panjang atau ukuran = 2. Panjang kesamaan karakter prefix yang juga merupakan suffiks ini dinamakan *Border Function* atau b(k). Jadi jumlah maksimum pergeseran yaitu pada contoh ini :

j-1 – b(k) = 5-2 = 3

4. Setelah ditemukan jumlah maksimum pergeseran (pada contoh diatas yaitu 3) selanjutnya ubah nilai j menjadi jumlah maksimum pergeseran. Karakter sebelum j akan pasti sama antara pattern dan teks karena karakter digeser sepanjang kesamaan dari prefix dan suffiks. Lanjutkan pemeriksaan dimulai dari nilai j yang baru. Lakukan pemeriksaan sampai semua karakter di teks sudah diperiksa atau berhasil didapatkan pattern yang bersesuaian.

Algoritma ini bagus apabila teks dan pattern mempunyai jenis/alphabet yang sedikit. Algoritma Knuth Morris Patt lebih baik dari algoritma brute force.

C. Algoritma Booyer Moore

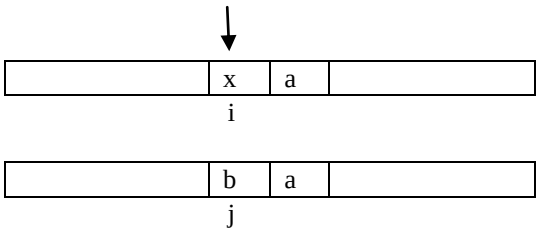
Algoritma Booyer Moore didasarkan pada 2 teknik, yaitu teknik *looking glass* dan teknik *character jump*.

1. Teknik *looking glass*

   Menemukan pattern di teks dimana pemeriksaan dimulai dari karakter paling akhir pattern dengan karakter yang bersesuaian pada teks.

2. Teknik *character jump*

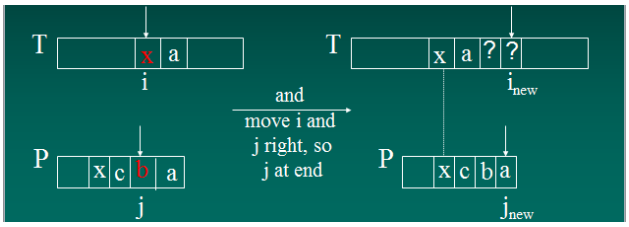
   Teknik dilakukan apabila terjadi *mismatch* atau ketidaksamaan string. Misalnya karakter T[i] tidak samadengan P[j].



Gambar 4. Ilustrasi Algoritma Booyer Moore

Ada 3 kemungkinan kasus pada Algoritma Booyer Moore, yaitu sebagai berikut.

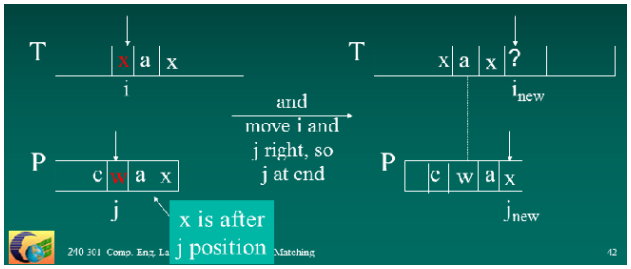
- Kasus 1



Gambar 5. Ilustrasi kasus 1 Algoritma Booyer Moore

Awalnya pemeriksaan dilakukan dari karakter akhir pattern (pada contoh diatas yaitu “a”). Pemeriksaan dilanjutkan sampai karakter awal pattern. Jika terjadi ketidaksamaan antara T[i] dengan P[j] kemudian cari *last occurent* (kemunculan terakhir) karakter di pattern yang sama dengan karakter di teks. Geser karakter sampai karakter yang sama tepat sejajar. Selanjutnya pindahkan j ke karakter akhir terbaru setelah pergeseran dan sesuaikan nilai i sama dengan j sehingga i dan j terletak sejajar.

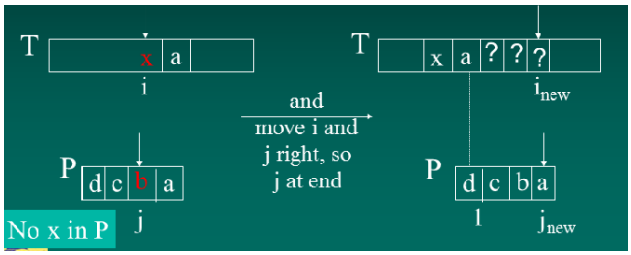
- Kasus 2



Gambar 5. Ilustarasi kasus 2 algoritma Booyer Moore

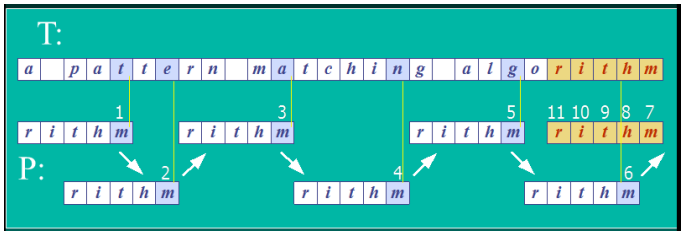
Kasus ini terjadi apabila saat pemeriksaan T[i] dan P[j] terjadi *mismatch* atau ketidaksamaan string dan *last occurent* nya sudah diperiksa pada tahapan sebelumnya. Hal ini diantisipasi dengan menggeser setiap karakter di pattern sebanyak 1 langkah. Setelah itu pindahkan nilai j ke karakter terkahir yang baru dan sesuaikan dengan nilai i sehingga posisi i dan j sejajar. Kemudian lakukan pemeriksaan kembali dari posisi paling akhir.

- Kasus 3



Gambar 9. Ilustrasi kasus 3 Algoritma Booyer Moore

Kasus ini terjadi apabila terjadi *mismatch* namun tidak ada *last occurent* di karakter yang akan diperiksa. Hal ini diatasi dengan menggeser pattern sehingga karakter awal pattern akan sejajar dengan karakter sesudah *mismatch* (pada gambar diatas *mismatch* terjadi dikarakter x, kemudian pattern digeser sampai karakter sejajar dengan karakter sesudah x). Berikut simulasi algoritma Booyer Moore



Gambar 10. Simulasi algoritma Booyer Moore

Algoritma Booyer Moore mempunyai *last occurent function* (fungsi kemunculan terakhir) yang akan dibaca saat pencocokan string dilakukan. Setiap karakter pada pattern dibuatkan indeks kemunculan terakhirnya didalam suatu *array* atau table. Algoritma Booyer Moore bagus untuk kasus dimana alfabet yang digunakan bervariasi, terutama untuk pencarian dalam teks berbahasa inggris.

III. PATTERN MATCHING PADA SIMSIMI

1. Pattern matching menjawab pertanyaan

Banyak yang mempertanyakan apakah SimSimi yang menjalankannya operator (manusia) atau benar-benar mesin karena obrolan SimSimi dengan manusia mudah dimengerti. Sebenarnya,antara pertanyaan dan jawaban pengguna sudah disimpan sebelumnya didatabase, sehingga pada saat pengguna lain mengajukan pertanyaan, sistem kemudian akan memeriksa apakah pertanyaan yang diberikan ada pada database sistem atau tidak. Pemeriksaan ini bisa dilakukan pengguna dengan pattern matching /pencocokan string. Misalnya pengguna memasukkan pertanyan :

"kamu lagi ngapain?"

Pencocokan pertanyaan pengguna dengan database pada SimSimi dapat dilakukan dengan berbagai algoritma, seperti algoritma brute force, algoritma knuth Moorish patt dan algoritma booyer moore. Pertanyaan berperan sebagai pattern, dibawah ini akan diberikan contoh pencocokan pertanyaan pengguna dengan pertanyaan yang terdapat di dalam database SimSimi.

1. Algoritma brute force

   Pencarian dilakukan didalam tabel database per record dengan algoritma brute force.

   Contoh :

   Salah satu record di database SimSimi :

k	a	m	u		u	d	a	h	
1	2	3	4	5	6				
k	a	m	u		l	a	g	i	

   Karena pada pemeriksaan ke-5 karakter pada database tidak sama dengan pattern pertanyaan pengguna, maka pattern digeser

k	a	m	u		u	d	a	h	

	k	a	m	u		l	a	g	i

k	a	m	u		u	d	a	h	

		k	a	m	u		l	a	g	i

k	a	m	u		u	d	a	h	

k	a	m	u		l	a	g	i

k	a	m	u		u	d	a	h	

	k	a	m	u		l

   Dan seterusnya sampai semua teks sudah diperiksa dengan algoritma brute force.

2. Algoritma knuth Moorish patt

   Pencarian yang dilakukan dengan algoritma knuth Moorish patt dilakukan dengan sebagai berikut :

   Contoh :

   Salah satu record dalam database SimSimi:

k	a	m	u		u	d	a	h	
1	2	3	4	5	6				
k	a	m	u		l	a	g	i	

Pada pemeriksaan ke-6 terjadi ketidakcocokan string. Selanjutnya diperiksa fungsi batas/ border *function*-nya.

Karena pada kasus diatas fungsi batas bernilai 0, maka untuk penggeseran pattern selanjutnya digeser sebanyak j-1 – b(k) = 5-0 = 5 sehingga menjadi :

k	a	m	u		u	d	a	h	
					7	8	9	10	
					k	a	m	u	

k	a	m	u

Dan seterusnya sampai semua teks diperiksa dengan algoritma Knutt Moorish Patt. Pencarian dengan algoritma Knutt Moorish Patt lebih efektif dari algoritma brute force karena proses pergeseran karakternya bisa lebih banyak dari pada algoritma brute force.

3. Algoritma Booyer Moore

Pencarian dilakukan dengan memeriksa kesamaan dari karakter paling akhir pattern.

Contoh :

Salah satu record didatabase SimSimi :

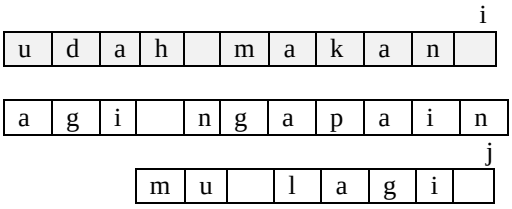
									i
u	d	a	h		m	a	k	a	n
								|	1
g	i		n	g	a	p	a	i	n
8	9		10	11	12	13	14	15	16j

Awalnya j berada pada karakter akhir pattern dan i sejajar dengan j. Kemudian diperiksa kesamaan antara T[i] dengan P[j]. Pada pemeriksaan kedua, dimana j=15 dan i=15, terjadi *mismatch*/ ketidakcocokan string pada pattern dengan teks. Sebelumnya dibuat daftar *last occurent* dari pattern (diasumsikan acuan 1 dari potongan pattern / dimulai dari karakter "g")

x	g	i	n	a	p
l(x)	11	15	16	14	13

Selanjutnya diperiksa *last occurent* nya, apabila karakter pada teks ('a') *last occurent* nya pada pattern

lebih kecil dari indeks sekarang yang sedang diperiksa, geser pattern sehingga karakter (‘a”) sejajar dengan ‘a’ pada pattern, seperti contoh dibawah:



Setelah pattern digeser, didapatkan i dan j yang baru, j bernilai posisi indeks paling akhir pattern. i yang baru terletak sejajar dengan j. Selanjutnya dilakukan lagi pemeriksaan dengan algoritma Booyer Moore sampai semua karakter pada teks sudah diperiksa.

Apabila setelah dilakukan pencarian di setiap record table pertanyaan dan ternyata pattern pertanyaan ada didalam database sistem SimSimi, sistem SimSimi menampilkan jawaban ke layar sesuai dengan jawaban yang tersimpan di database untuk pertanyaan tersebut. Apabila tidak ditemukan kesamaan antara pertanyaan pengguna dan pertanyaan yang ada didalam sistem, sistem SimSimi akan menampilkan pesan “I have no respon. Please teach me”. Pengguna dapat mengajarkan pertanyaan baru berikut jawaban terbarunya. Pertanyaan yang diajarkan mungkin sudah pernah diajarkan dengan pertanyaan sebelumnya, itu berarti akan jawaban yang diberikan pengguna akan disimpan sebagai jawaban berikutnya oleh database. Dalam memeriksa apakah pertanyaan sudah ada didatabase atau tidak dapat digunakan pattern matching. SimSimi dapat menampilkan berbagai jawaban, sesuai dengan berbagai jawaban di databases sistem SimSimi

Gambar. Percakapan dengan pertanyaan yang sama

2. Pattern matching dalam menyeleksi kalimat

Dalam mengajari SimSimi menjawab pertanyaan, tak jarang pertanyaan dan jawaban yang diberikan pengguna mengandung kata-kata yang tidak sopan. Dari percobaan percakapan dengan SimSimi pun masih ditemukan kata-kata tidak sopan yang diajarkan oleh pengguna usil dan tersimpan di database sistem.

Pattern matching juga dapat digunakan dalam penyeleksian pertanyaan dan jawaban yang diajarkan pengguna, sehingga pertanyaan dan jawaban yang disimpan tidak mengandung kata-kata tidak sopan. Hal ini dapat dilakukan dengan membuat database yang berisi kata-kata yang tidak sopan. Setelah pengguna memilih tombol “teach” sistem kemudian akan memeriksa apakah dalam pertanyaan dan jawaban yang diajarkan mengandung kata-kata kotor atau tidak. Pemeriksaan ini dapat meilbatkan string matching, memeriksan pertanyaan dan jawaban dari pengguna sebagai pattern dan memeriksa kesamaannya dengan database sitem Simisimi. Jika mengandung kata-kata kotor, kalimat tersebut bisa tidak ditampilkan ke layar.

Gambar. Pengaturan pemilihan kata

IV. KESIMPULAN

Pattern matching banyak digunakan dalam berbagai aplikasi,seperti pencocokan *string* yang akan memeriksa kesamaan *pattern* string dengan kumpulan string/teks. Untuk memeriksa kesamaan ini, ada banyak algoritma yang bisa digunakan, diantaranya yaitu algoritma Brute Froce, algoritma Knutt Moorish Patt dan algoritma Booyer Moore. Ketiganya memiliki ciri khas masing-masing dan cocok diimplementasikan sesuai dengan jenis text yang digunakan.

Seiring dengan perkembangan teknologi, saat ini banyak muncul aplikasi baru yang bisa memenuhi berbagai kebutuhan manusia, salah satunya aplikasi

chatting dengan robot/mesin SimSimi. Aplikasi ini mengimplementasikan string matching dalam pencocokan pertanyaan yang akan diberikan pengguna dengan pertanyaan yang jawabannya sudah tersimpan di database sistem SimSimi. Dalam proses pencocokannya kita dapat mengimplementasikan algoritma Brute force, algoritma Knutt Moorish Patt dan algoritma Booyer moore. Selain itu, string matching juga dapat digunakan untuk menyeleksi pertanyaan dan jawaban yang diberikan pengguna.

V. REFERENSI

[1] http://www.simsimi.com/talk.htm
[2] http://sekarsekir.blogspot.com/2012/06/percakapan-dengan-simsimi.html
[3] http://dailysocial.net/post/simsimi-membantu-anda-mengisi-waktu-luang

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah hasil tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 18 Mei 2014

Diah Fauziah /13512049