Penerapan Algoritma Squeezer untuk Memberikan Rekomendasi Pilihan Lagu Berdasarkan Daftar Lagu yang Dimainkan pada Pemutar Mp3 Android

Eko Wahyu Wibowo, Siti Rochimah, dan Abdul Munif.
Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111
E-mail: siti_r@yahoo.com

***Abstrak*—Pada tulisan ini dibuat sebuah aplikasi pemutar mp3 pada perangkat Android dan juga layanan web yang terintegrasi menjadi sebuah sistem yang dapat memberikan rekomendasi lagu kepada penggunanya. Rekomendasi yang diberikan akan didasarkan pada riwayat lagu yang sering didengarkan oleh pengguna. Untuk menghasilkan rekomendasi digunakan algoritma Squeezer. Algoritma Squeezer membantu dalam mengelompokkan riwayat lagu pengguna dan memberikan rekomendasi lagu baru yang mirip dengan kelompok riwayat lagu yang paling besar. Berdasarkan hasil pengujian yang dilakukan tingkat akurasi rekomendasi yang diberikan mencapai 80% serta mendapat respon yang baik dari pengguna.**

***Kata Kunci*—Android, lagu, rekomendasi, Squeezer.**

I. PENDAHULUAN

Sistem rekomendasi merupakan sebuah sistem yang dapat memberikan rekomendasi kepada pengguna sistem. Rekomendasi yang diberikan dapat berdasarkan karakteristik data personal pengguna atau lingkungan sosial pengguna tersebut.Saat ini penerapan dari sistem rekomendasi sudah banyak dilakukan dengan menggunakan berbagai macam metode dan pendekatan [1].

Sudah tidak diragukan lagi jika musik merupakan simfoni kehidupan, musik tidak hanya sebagai hiburan semata melainkan juga mampu memberikan energi yang berbeda pada pendengarnya. Kebutuhan akan musik menjadi penting, yang ditunjukkan dengan banyaknya pelaku yang terlibat dalam industri ini. Dengan menerapkan sistem rekomendasi tersebut pada industri ini akan membantu pendengar untuk memilih musik yang akan didengarkannya. Selain itu juga membantu produsen memasarkan musiknya pada pendengar yang tepat.

Untuk itu, pada tulisan ini dibuat sebuah aplikasi pemutar mp3 pada perangkat Android dan juga layanan web yang terintegrasi menjadi sebuah sistem yang dapat memberikan rekomendasi lagu kepada penggunanya. Rekomendasi yang diberikan akan didasarkan pada riwayat lagu yang sering didengarkan oleh pengguna.

Untuk menghasilkan rekomendasi dari riwayat lagu tersebut digunakan algoritma Squeezer. Algoritma tersebut akan membantu dalam mengelompokkan riwayat lagu pengguna dan memberikan rekomendasi lagu baru yang mirip dengan kelompok riwayat lagu yang paling besar. Rekomendasi yang diberikan cukup akurat. Berdasarkan hasil pengujian yang dilakukan tingkat akurasi rekomendasi yang diberikan mencapai 80% serta mendapat respon yang baik dari pengguna.

II. METODE DAN PERANCANGAN SISTEM

A. Algoritma Squeezer

Squeezer merupakan sebuah algoritma yang digunakan untuk mengelompokkan data (*clustering*) sekumpulan data bertipe kategorikal. Ide dasar dari algoritma tersebut sangat sederhana. Squeezer secara berulang membaca tiap pasangan data (*tuple*) dari kumpulan data satu persatu. Saat pasangan data pertama dibaca, akan dibuat kelompok data (*cluster*) baru. Pasangan data berikutnya dimasukkan kedalam kelompok data yang sudah ada atau ditolak oleh semua kelompok yang ada sehingga membentuk kelompok baru berdasarkan fungsi kemiripan yang diberikan antara kelompok dengan pasangan data [2].

Berikut adalah definisi yang digunakan oleh algoritma Squeezer. Misal $A_1, ..., A_m$ adalah himpunan kategorikal atribut dengan domain $D_1, ...,D_m$ berturut-turut. Misal himpunan data D adalah himpunan dari pasangan data dimana setiap pasangan data $t : t \in D_1 x ... x D_m$. Misal *TID* adalah himpunan dari *ID* unik dari setiap pasangan data. Untuk setiap $tid \in TID$, nilai atribut A_1 dari pasangan datayang bersangkutan direpresentasikan sebagai $val(tid, A_i)$ [2].

Definisi 1(*Kelompok*). *Kelompok* = $\{tid \mid tid \in TID\}$ *adalah himpunan bagian dari TID.* (1)

Definisi 2. *Diberikan Kelompok C, himpunan nilai atribut pada A_i yang ada pada C didefinisikan sebagai:*
$VAL_i(C) = \{val(tid, A_i) | tid \in C\}$. (2)

Definisi 3. *Diberikan Kelompok C, dan $a_i \in D_i$ maka dukungan (jumlah tuple pada cluster yang memiliki nilai tersebut) dari a_i pada C dengan A_i yang terkait, didefinisikan sebagai:*
$Sup(a_i) = |\{tid | tid.A_i = a_i\}|$. (3)

Definisi 4 (*Rangkuman*). *Diberikan Kelompok C, Rangkuman (Summary) untuk C didefinisikan sebagai:*

Algorithm Squeezer(*D*,*s*)
Begin
1. while (*D* has unread tuple){
2. tuple = getCurrentTuple(*D*)
3. if(tuple.tid == 1){
4. *addNewClusterStructure*(*tuple*, *tid*)}
5. else{
6. for each existing cluster *C*
7. *simComputation*(*C*, *tuple*)
8. get the max value of similarity : *sim_max*
9. get the corresponding Cluster Index: *index*
10. if *sim_max*>= *s*
11. *addTupleToCluster*(*tuple*, *index*)
12. else
13. *addNewClusterStructure*(*tuple*, *tid*)}
14. }
15. *handleOutliers*()
16. *outputClusteringResult*()

End

Gambar 1. Algoritma Squeezer [2]

Sub_Function*addNewClusterStructure*(*tid*)

1. *Cluster* = {*tid*}
2. for each attribute value a_1 on A_i
3. VS_i= (a_i,1)
4. *add* $VS_i$to *Summary*
5. *CS* = {*Cluster*, *Summary*}

Gambar 2. Sub-fungsi *addNewClusterStructure()* [2]

Sub_Function*addTupleToCluster*(*tuple*, *index*)

1. *Cluster* = *Cluster* U{*tuple*, *tid*}
2. for each attribute value a_1 on A_i
3. VS_i= (a_i,*Sup*(a_i)+1)
4. *add* $VS_i$to *Summary*
5. *CS* = {*Cluster*, *Summary*}

Gambar 3. Sub-fungsi *addTupleToCluster()* [2]

Sub_Function*simComputation*(*C*, *index*)

1. Defin *sim* = 0
2. for each attribute value a_1 on A_i
3. *sim* = *sim* + *probability of* a_i *on C*
4. return *sim*

Gambar 4. Sub-fungsi *simComputation()* [2]

Rangkuman = $\{VS_i | 1 \leq i \leq m\}$ *dimana* VS_i = $\{(a_i, Sup(a_i)) | a_i \in VAL_i(C)\}$. (4)

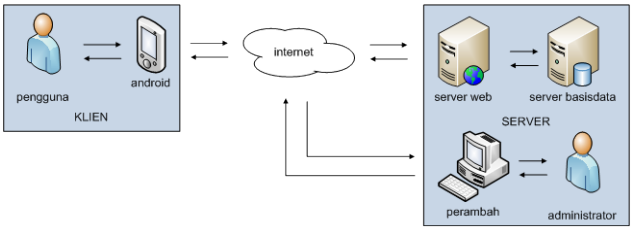
Definisi 5. *Diberikan Kelompok C, Struktur Kelompok (Cluster Structure (CS)) dari C didefiniskan sebagai:*
$CS = \{C, Rangkuman\}$. (5)

Definisi 6 (Fungsi Kemiripan). *Diberikan Kelompok C dan sebuah pasngan data t dengan tid* ∈ *TID. Kemiripan antara t dengan C didefinisikan sebagai:*
$Sim(C,tid) = \sum_{i=1}^{m}(\frac{Sup(a_i)}{\sum_j Sup(a_j)})$ *dimana tid.*$A_i = a_i$ *dan* a_j ∈$VAL_i(C)$. (6)

Gambar 1 sampai dengan Gambar 4 menjelaskan penerapan algoritma Squeezer secara umum.

B. Deskripsi Umum Sistem

Sistem yang dibuat adalah sebuah sistem yang mampu memberikan rekomendasi lagu kepada pengguna.



Gambar 5. Deskripsi umum sistem

Rekomendasi yang diberikan didasarkan pada daftar lagu yang dimainkan oleh pengguna. Secara umum gambaran alur sistem dapat dilihat pada Gambar 5.

Pada Gambar 5 sistem terdiri dari dua bagian, yaitu bagian klien dan bagian server. Bagian klien dalam bentuk aplikasi pemutar mp3 pada perangkat Android yang bertugas untuk merekam riwayat lagu dan mengirimkannya ke server [3]. Bagian server dalam bentuk layanan web dan halaman admin. Server sendiri bertugas untuk mengolah data dan memberikan rekomendasi. Komunikasi antara server dengan klien dilakukan melalui layanan web, sedangkan untuk mengatur data oleh administrator dapat dilakukan melalui halaman admin [4].

Pada sistem ini pendekatan yang digunakan untuk memberikan rekomendasi lagu adalah dengan mengelompokkan lagu yang dimainkan pada pemutar mp3 untuk mewakili selera musik pengguna. Dari kelompok tersebut akan diambil kelompok dengan jumlah lagu terbanyak dan dianggap mewakili selera musik pengguna. Kemudian dari kelompok lagu tersebutakan diberikan rekomendasi lagu yang mirip dengan kelompok tersebut. Secara sederhana pendekatan ini merupakan pendekatan penyaringan berbasis konten dimana pengenalan tag lagu akan sangat berpengaruh terhadap hasil rekomendasi yang diberikan [5].

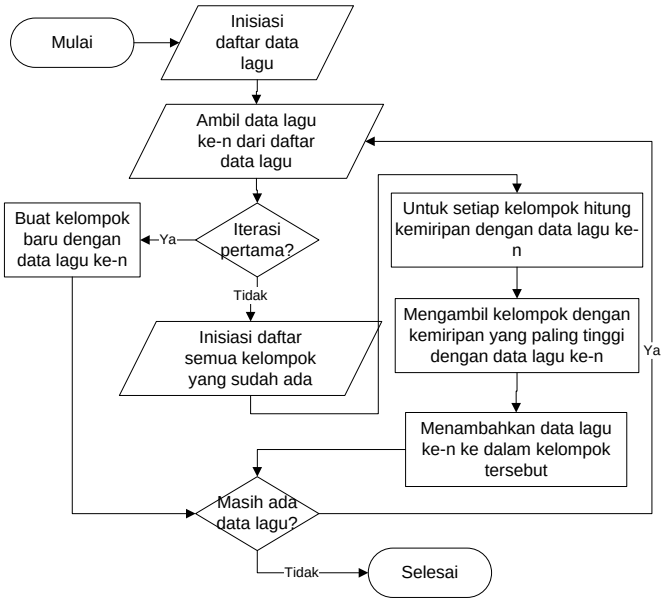
Algoritma Squeezer membantu mengelompokkan daftar lagu berdasarkan tag-tag yang ada dalam lagu tersebut, misalnya artis, album atau genre. Pembobotan pada setiap tag untuk menentukkan pengelompokan juga dapat dilakukan, sehingga pengguna dapat menyesuaikan rekomendasi yang diberikan.

Rekomendasi yang diterima oleh pengguna merupakan daftar lagu yang belum pernah didengarkan dan mirip dengan lagu yang sering didengarkan oleh pengguna tersebut. Sistem juga akan membantu pengguna untuk mencari informasi lagu hasil rekomendasi tersebut melalui mesin pencari.

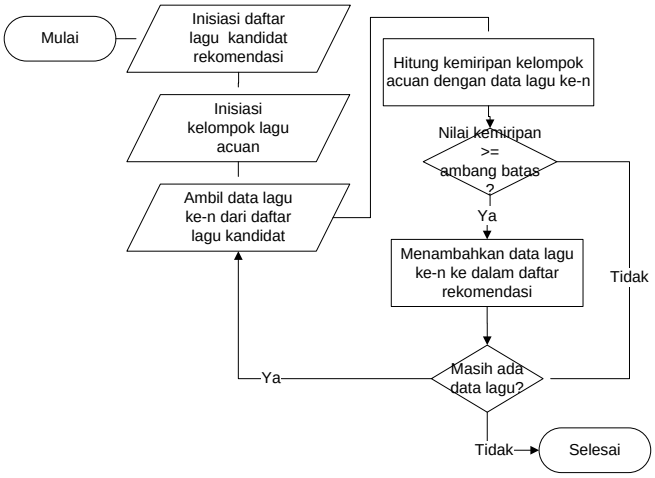
C. Perancangan Implementasi Algoritma Squeezer pada Sistem

Untuk dapat mengimplementasikan algoritma Squeezer sebagai alat dalam mengelompokkan lagu dan dasar rekomendasi, perlu dilakukan beberapa penyesuaian. Sebelum membahas rancangan persamaan yang akan digunakan, berikut merupakan alur proses pengelompokan lagu yang digunakan. Alur proses tersebut digambarkan dengan diagram alir seperti yang terlihat pada Gambar 6. Kemudian pada Gambar 7 merupakan diagram alir proses bagaimana rekomendasi diberikan.

Jika diperhatikan dengan seksama, kedua alur proses tersebut sebenarnya merupakan alur algoritma Squeezer itu sendiri. Hanya saja yang berbeda adalah data yang digunakan.



Gambar 6. Diagram alir proses pengelompokkan lagu

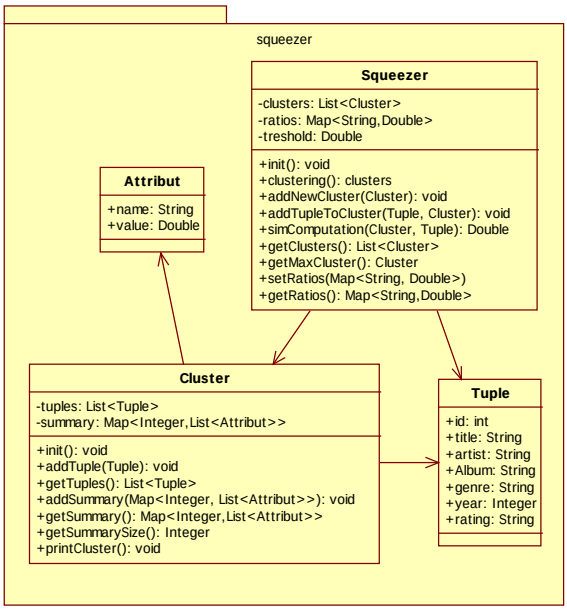


Gambar 7. Diagram alir proses pemberian rekomendasi

Algoritma Squeezer dirancang untuk data kategorikal secara umum, sedangkan pada sistem ini digunakan data dari tag-tag lagu. Selain itu pada algoritma Squeezer tidak ada pembobotan rasio pada masing-masing kategori. Sedangkan pada sistem yang dibuat dimungkinkan untuk melakukan hal tersebut. Untuk menyesuaikan hal tersebut maka persamaan dari algoritma yang sudah ada akan mengalami beberapa penyesuaian.

Seperti yang terlihat pada diagram alir pada Gambar 6 dan 7, persamaan yang dipakai pada keduanya adalah persamaan untuk menghitung kemiripan antara data lagu dengan kelompok lagu. Persamaan tersebut merupakan persamaan kemiripan algortima Squeezer yang dijelaskan pada bagian II.A. Sehingga persamaan itulah yang perlu disesuaikan terhadap format data lagu yang digunakan.

Setiap data lagu yang akan dikelompokkan memiliki tag ARTIS, ALBUM, GENRE, TAHUN dan RATING sebagai



Gambar 8.Diagram kelas algoritma Squeezer

atribut kategorikal. Dengan format data tersebut maka persamaan kemiripan diubah menjadi Persamaan 7.

Dari Persamaan7 didapatkan kemiripan antara data sebuah lagu dengan kelompok lagu acuan. Selain itu pada persamaan tersebut akan didapatkan pembobotan untuk masing-masing atribut.

$$\text{Kemiripan}(n,k) = r1\ X\ \frac{\text{jumlah ARTIS n yang sama pada k}}{\text{jumlah anggota k}} + r2\ X\ \frac{\text{jumlah ALBUM n yang sama pada k}}{\text{jumlah anggota k}} + r3\ X\ \frac{\text{jumlah GENRE n yang sama pada k}}{\text{jumlah anggota k}} + r4\ X\ \frac{\text{jumlah TAHUN n yang sama pada k}}{\text{jumlah anggota k}} + r5\ X\ \frac{\text{jumlah RATING n yang sama pada k}}{\text{jumlah anggota k}} \quad (7)$$

Keterangan : n = data lagu yang diuji kemiripannya
k = kelompok lagu acuan
r1 = rasio bobot atribut ARTIS
r2 = rasio bobot atribut ALBUM
r3 = rasio bobot atribut GENRE
r4 = rasio bobot atribut TAHUN
r5 = rasio bobot atribut RATING
r1+r2+r3+r4+r5 = 1

Persamaan 7 inilah yang diimplementasikan pada sistem sebagai dasar untuk menghitung kemiripan. Dengan menggunakan rasio pada masing-masing atribut maka nilai kemiripan dapat diatur atau dititiberatkan ke atribut-atribut tertentu. Selain itu dengan membuat jumlah seluruh rasio sama dengan 1 membuat nilai kemiripan dapat dijaga jangkauannya yaitu antara 0 sampai 1.

D. Diagram Kelas Algoritma Squeezer

Gambar 8 menunjukkan diagram kelas yang digunakan untuk mengimplementasikan algoritma Squeezer. Diagram kelas tersebut dibuat berdasarkan penjelasan yang ada pada bagian II.C. Kelas `Squeezer` merupakan kelas pengendali utama untuk menggunakan algoritma tersebut. Kelas tersebut

berfungsi untuk melakukan pengelompokan data dan juga untuk mendapatkan kelompok data yang paling besar. Data yang dikelompokkan di dalam kelas diagram tersebut diwakili oleh kelas `Tuple`. Kemudian kelompok data yang dibentuk diwakili oleh kelas `Cluster`.

III. IMPLEMENTASI

A. *Implementasi Algoritma Squeezer*

Implementasi algoritma Squeezer dilakukan pada bagian server dan mengacu pada penjelasan bagian II.C dan II.E tentang algoritma tersebut.Berikut merupakan kelas-kelas yang dibutuhkan untuk mengimplementasikan algoritma tersebut.

1) *Kelas* `Tuple`

Kelas `Tuple` merupakan wujud dari pasangan data yang digunakan dalam algoritma Squeezer. Pasangan data yang digunakan adalah data lagu, sehingga kelas `Tuple` merepresentasikan objek lagu. Seperti halnya model `Song` atau lagu, kelas ini juga memiliki atribut yang sama. Atribut tersebut yang akan digunakan dalam proses pengelompokkan data.

2) *Kelas* `Attribute`

Kelas `Attribute` merupakan kelas bantuan untuk menyimpan nilai atau bobot dari setiap atribut yang ada pada kelas `Tuple`.

3) *Kelas* `Cluster`

Kelas `Cluster` merupakan representasi kelompok yang digunakan dalam algoritma Squeezer. Fungsi yang dimiliki kelas ini adalah menambahkan `Summary` ke dalam `Cluster` tersebut. Selain fungsi tersebut kelas ini memang dibuat untuk mendukung jalannya algoritma Squeezer.

4) *Kelas* `Squeezer`

Kelas `Squeezer` merupakan kelas pengendali utama dari algoritma ini dan sebagai wujud dari algoritma itu sendiri. Kelas ini memiliki fungsi utama yaitu fungsi untuk mengelompokkan data berdasarkan algoritma Squeezer. Berikut fungsi-fungsi yang dipakai.

a) *Fungsi Clustering*

Merupakan fungsi utama pada kelas `Squeezer`. Fungsi ini melakukan pengelompokan data yang berupa kelas `Tuple` ke dalam kelompok-kelompok dalam bentuk kelas `Cluster`. Fungsi ini bisa juga disebut algoritma Squeezer itu sendiri. Fungsi ini mengimplementasikan kode semu (*pseudo code*) yang ditunjukkan Gambar 1. Seperti yang dijelaskan pada kode semu, masukan fungsi ini adalah himpunan data yang akan dikelompokkan beserta konfigurasinya. Kembalian dari fungsi ini adalah daftar kelompok-kelompok yang terbentuk beserta anggotanya. Implementasi fungsi ini ditunjukkan Gambar 9.

```
....
public ArrayList<Cluster> clustering(Float treshold,
ArrayList<Tuple> dataset){
this.treshold = treshold;
this.dataset = dataset;
        Float maxSim;
        Float sim;
int maxId;
int id;
for(Tuple tuple : this.dataset){
            maxSim  = new Float(0);
            sim     = new Float(0);
            maxId   = -1;
            id      = 0;
for(Cluster cluster : this.clusters){
                sim = simComputation(cluster, tuple);
if(sim > maxSim){
                    maxSim  = sim;
                    maxId   = id;
                }
                id++;
            }
if(maxSim >= treshold) {
        addTupleToCluster(tuple, this.clusters.get(maxId));
            }
else {
                addNewCluster(tuple);
            }
        }
returnthis.clusters;
    }
....
```

Gambar 9. Kode sumber fungsi `clustering`

Pada Gambar 9 tersebut terlihat bagaimana data lagu yang ada akan dikelompokkan. Pengelompokan tersebut didasarkan pada kemiripan data lagu dengan kelompok-kelompok yang sudah ada. Jika kemiripan yang dihasilkan lebih besar atau sama dengan ambang batas yang ditentukan maka data lagu tersebut dapat menjadi anggota kelompok yang sudah ada, jika tidak maka akan dibuat kelompok baru dengan anggota data lagu tersebut.

b) *Fungsi simComputation*

Bisa disebut juga sebagai persamaan kemiripan yang telah dijelaskan bagian II.C. Pada implementasinya fungsi ini merupakan fungsi untuk menghitung kemiripan antara suatu kelas `Tuple` dengan kelas `Cluster` yang ada. Fungsi ini juga yang digunakan untuk menentukan rekomendasi lagu yang mirip dengan `Cluster` yang mewakili selera pengguna atau `Cluster` dengan anggota terbanyak. Fungsi ini mengimplementasikan kode semu yang ditunjukkan Gambar 4.Seperti yang dijelaskan pada kode semu, masukan fungsi ini adalah `Cluster` dan `Tuple` yang akan dicek kemiripannya. Kembalian dari fungsi ini adalah nilai kemiripan dari `Tuple` tersebut. Rincian impelementasi fungsi iniditunjukkan pada Gambar 10.

IV. PENGUJIAN DAN EVALUASI

Pengujian dilakukan untuk menguji keakurasian rekomendasi yang diberikan.Keakurasian ini didasarkan pada konfigurasi dan daftar lagu yang ada pada riwayat lagu pengguna. Pengujian ini akan dilakukan dalam beberapa skenario. Setiap skenarioakan menguji keakurasian rekomendasi yang diberikan. Rincian terkait masing-masing skenario dapat dilihat pada Tabel 1.

```
...
public Float simComputation(Cluster cluster, Tuple tuple){
        Float sim = new Float(0);
            sim +=
ratios.get(Cluster.ALBUM) *(probAttribut(tuple.getAlbum(),
cluster.getSummary(Cluster.ALBUM)) /
cluster.getSummarySize(Cluster.ALBUM));
            sim +=
ratios.get(Cluster.ARTIST)*(probAttribut(tuple.getArtist(),
cluster.getSummary(Cluster.ARTIST)) /
cluster.getSummarySize(Cluster.ARTIST));
            sim +=
ratios.get(Cluster.GENRE)*(probAttribut(tuple.getGenre(),
cluster.getSummary(Cluster.GENRE)) /
cluster.getSummarySize(Cluster.GENRE));
            sim +=
ratios.get(Cluster.YEAR)*(probAttribut(tuple.getYear().toSt
ring(), cluster.getSummary(Cluster.YEAR)) /
cluster.getSummarySize(Cluster.YEAR));
            sim +=
ratios.get(Cluster.RATING)*(probAttribut(tuple.getRating(),
cluster.getSummary(Cluster.RATING)) /
cluster.getSummarySize(Cluster.RATING));
return sim;
    }
....
```

Gambar 10. Kode sumber fungsi `simComputation`

Tabel 1.
Skenario Pengujian

Skenario	Jumlah riwayat lagu	Konfigurasi
Skenario 1	50	Ambang batas : 0.7 Artis : 0.2 Album : 0.2 Genre : 0.2 Tahun : 0.2 Peringkat : 0.2
Skenario 2	50	Ambang batas : 0.6 Artis : 0.4 Album : 0.0 Genre : 0.4 Tahun : 0.1 Peringkat : 0.1

Tabel 2.
Kelompok Lagu Terbesar Skenario 1

No	Judul	Artis	Album	Genre	Tahun	Rating
1	Your Call	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
2	Fall For You	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
3	Maybe	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
4	Your Call	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
5	Fall For You	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
6	Maybe	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
7	Maybe	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
8	Fall For You	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome

Tabel 3.
Rekomendasi Yang Diberikan Skenario 1

No	Judul	Artis	Album	Genre	Tahun
1	Like A Knife	Secondhand Serenade	A Twist In My Story	Pop	2008
2	Stranger	Secondhand Serenade	A Twist In My Story	Pop	2008
3	Suppose	Secondhand Serenade	A Twist In My Story	Pop	2008
4	A Twist In My Story	Secondhand Serenade	A Twist In My Story	Pop	2008
5	Why	Secondhand Serenade	A Twist In My Story	Pop	2008
6	Pretend	Secondhand Serenade	A Twist In My Story	Pop	2008
7	Stay Close, Don't Go	Secondhand Serenade	A Twist In My Story	Pop	2008
8	Goodbye	Secondhand Serenade	A Twist In My Story	Pop	2008

Tabel 4.
Kelompok Lagu Terbesar Skenario 2

No	Judul	Artis	Album	Genre	Tahun	Rating
1	Muak	Andra and The Backbone	Season 2	Pop	2008	Awesome
2	Muak	Andra and The Backbone	Season 2	Pop	2008	Awesome
3	Muak	Andra and The Backbone	Season 2	Pop	2008	Awesome
4	Nothings on you	Bruno Mars	Bruno	Pop	2008	Awesome
5	Payphone	Maroon 5	Maroon 5	Pop	2008	Awesome
6	Payphone	Maroon 5	Maroon 5	Pop	2008	Awesome
7	Payphone	Maroon 5	Maroon 5	Pop	2008	Awesome
8	Payphone	Maroon 5	Maroon 5	Pop	2008	Awesome
9	Your Call	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
10	Fall For You	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
11	Maybe	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
12	Your Call	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
13	Fall For You	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
14	Maybe	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
15	Maybe	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome
16	Fall For You	Secondhand Serenade	A Twist In My Story	Pop	2008	Awesome

Tabel 5.
Rekomendasi Yang Diberikan Skenario 2

No	Judul	Artis	Album	Genre	Tahun
1	Like A Knife	Secondhand Serenade	A Twist In My Story	Pop	2008
2	Stranger	Secondhand Serenade	A Twist In My Story	Pop	2008
3	Suppose	Secondhand Serenade	A Twist In My Story	Pop	2008
4	A Twist In My Story	Secondhand Serenade	A Twist In My Story	Pop	2008
5	Why	Secondhand Serenade	A Twist In My Story	Pop	2008
6	Pretend	Secondhand Serenade	A Twist In My Story	Pop	2008
7	Stay Close, Don't Go	Secondhand Serenade	A Twist In My Story	Pop	2008
8	Goodbye	Secondhand Serenade	A Twist In My Story	Pop	2008
9	Kepayang	Andra and The Backbone	Season 2	Pop	2008
10	Main Hati	Andra and The Backbone	Season 2	Pop	2008
11	Mimpi Yang Terbunuh	Andra and The Backbone	Season 2	Pop	2008
12	Hitamku	Andra and The Backbone	Season 2	Pop	2008
13	Selamat Tinggal	Andra and The Backbone	Season 2	Pop	2008
14	Hot 'N Cold	Katy Perry	One of The Boys	Pop	2008
15	Goodnight Goodnight	Maroon 5	It Won't Be	Pop	2007
16	Kiwi	Maroon 5	It Won't Be	Pop	2007
17	Luka Ku	d'Masiv	Perubahan	Pop	2008
18	Merindukanmu	d'Masiv	PERUBAHAN	Pop	2008
19	Pemain Cinta	Ada Band	Che	Pop	2008
20	Baiknya	Ada Band	Harmonious	Pop	2008
21	She Will Be Loved	Maroon 5	Songs About Jane	Pop	2004

Tabel 6.
Kemiripan Rekomendasi Skenario 1

No	Judul rekomendasi	Kemiripan	Hasil
1	Like A Knife	1	Akurat
2	Stranger	1	Akurat
3	Suppose	1	Akurat
4	A Twist In My Story	1	Akurat
5	Why	1	Akurat
6	Pretend	1	Akurat
7	Stay Close, Don't Go	1	Akurat
8	Goodbye	1	Akurat
	Rata-rata kemiripan	1	-

Pada masing-masing skenario tersebut akan menggunakan *master* data lagu yang sama. Jumlah data lagu yang digunakan pada server berjumlah 300 lagu. Hasil pengujian dari masing-masing skenario dapat dilihat pada Tabel 2 sampai dengan Tabel 5.

Untuk menghitung akurasi pada skenario pengujian 1 dan 2 digunakan Persamaan 7. Tabel 6 dan 7 menunjukkan nilai kemiripan dari hasil perhitungan yang dilakukan.

Tabel 7.
Kemiripan Rekomendasi Skenario 2

No	Judul rekomendasi	Kemiripan	Hasil
1	Like A Knife	0.8	Akurat
2	Stranger	0.8	Akurat
3	Suppose	0.8	Akurat
4	A Twist In My Story	0.8	Akurat
5	Why	0.8	Akurat
6	Pretend	0.8	Akurat
7	Stay Close, Don't Go	0.8	Akurat
8	Goodbye	0.8	Akurat
9	Kepayang	0.67	Akurat
10	Main Hati	0.67	Akurat
11	Mimpi Yang Terbunuh	0.67	Akurat
12	Hitamku	0.67	Akurat
13	Selamat Tinggal	0.67	Akurat
14	Hot 'N Cold	0.6	Akurat
15	Goodnight Goodnight	0.6	Akurat
16	Kiwi	0.6	Akurat
17	Luka Ku	0.6	Akurat
18	Merindukanmu	0.6	Akurat
19	Pemain Cinta	0.6	Akurat
20	Baiknya	0.6	Akurat
21	She Will Be Loved	0.6	Akurat
	Rata-rata kemiripan	0.69	-

Berdasarkan hasil pengujian pada skenario 1 dan 2 didapatkan hasil bahwa seluruh rekomendasi yang diberikan akurat dengan rata-rata tingkat akurasi (1 + 0.69) / 2 = 0.845 atau 84 %. Tingkat keakurasian ini sangat dipengaruhi oleh data pada *master* data lagu server. Selain tingkat keakurasian yang dapat dihitung, pada pengujian ini juga didapatkan kesimpulan lain. Dengan menggunakan data riwayat lagu yang sama tetapi dengan konfigurasi yang berbeda akan menghasilkan rekomendasi yang berbeda. Sehingga dapat disimpulkan bahwa pengguna dapat mengatur rekomendasi yang diinginkan melalui konfigurasi tersebut dengan tingkat keakurasian yang cukup tinggi.

V. KESIMPULAN/RINGKASAN

Sistem dapat memberikan rekomendasi pilihan lagu kepada pengguna berdasarkan data riwayat lagu yang dimainkan pengguna. Algoritma Squeezer dapat diimplementasikan untuk memberikan rekomendasi lagu tersebut dengan tingkat akurasi yang cukup tinggi, yaitu 84% dengan 300 data *master* dan 50 data uji.

DAFTAR PUSTAKA

[1] R. Francesco, R. Lior, and S. Bracha. *Introduction to Recommender Systems Handbook, RecommenderSystems Handbook*. Amerika Serikat:Springer, (2011).

[2] Z. He, X. Xu, dan S. Deng, "Squeezer:An Efficient Algorithm for Clustering Categorical Data". *Journal of Computer Science and Technology*, Vol. 17 No.5, (2002) 611-624.

[3] R. Clarkson, *Spring for Android Reference Manual.* <URL: http://static.springsource.org/springandroid/docs/1.0.x/reference/htmlsingle diakses 23 April 2012>, (2010).

[4] D. Chappell dan T. Jewell, *Java Web Services.* Amerika Serikat:O'Reilly, (2002).

[5] Wikepedia. *Pandora Radio*.Wikipedia. <URL: http://en.wikipedia.org/wiki/Pandora_Radio diakses10 April 2012> (2012).