ISSN 2355-3286

Implementasi Algoritma Squeezer dan Term Frequency Ranking dalam Pembangunan Sistem Rekomendasi Tempat Makan

Vincentius Wirawan[1], Seng Hansun[1], Hargyo Tri Nugroho I.[2]
[1] Program Studi Teknik Informatika, Universitas Multimedia Nusantara, Tangerang, Indonesia
vincentius.wirawan@yahoo.com, hansun@umn.ac.id
[2] Program Studi Sistem Komputer, Universitas Multimedia Nusantara, Tangerang, Indonesia
hargyo@gmail.com

Diterima 15 Maret 2014
Disetujui 19 Juni 2014

***Abstract*—Food is something that can not be separated from the life of every human being. Its variation makes a lot of people interested in tasting many types of food, even they have not known yet. The detail information of restaurants, especially their location could be a problem. So, the restaurants recommendation website is made to help them. The recommendations are given by the Squeezer algorithm using the user's input as references. The accuracy of the given recommendations are determined from the amount and consistency of the user at the time of selecting the item. The Term Frequency Ranking is used as a searching feature to display the search results in ordered manner by their relevancy rank. Experiment result shows that the Term Frequency Ranking gives high value of Recall, 1, and 0.46 for the Precision value.**

***Index Terms*—Recommended System, Squeezer, Term Frequency Ranking, Website.**

I. PENDAHULUAN

Makanan merupakan suatu hal yang tidak dapat dilepaskan dari kehidupan setiap manusia. Keanekaragaman jenis makanan membuat banyak orang tertarik untuk mencicipi banyak jenis makanan, bahkan yang mereka belum ketahui. Kadang, tidak semua daerah memiliki semua jenis makanan. Ada beberapa daerah yang memiliki jenis makanan tertentu. Hal ini membuat para pecinta makanan berjelajah ke daerah-daerah tertentu untuk menikmati makanan-makanan yang berada di daerah tersebut.

Dalam hal ini, tentunya banyak orang yang tidak mengetahui informasi lengkap tentang tempat makan pada suatu daerah tersebut seperti jenis makanan, harga, bahkan lokasinya. Hal ini menjadi acuan dibuatnya banyak situs untuk berbagi informasi tentang tempat makan. Banyak orang dapat mencari dengan mudah informasi-informasi tentang makanan apa saja yang berada pada daerah yang dituju.

Akan tetapi, masih banyak situs-situs yang tidak memiliki sistem rekomendasi bagi pengunjung dalam memberikan rekomendasi makanan yang mungkin sejenis dan lebih baik, yang *user* belum tahu. Melalui keterbatasan ini, muncullah ide untuk membuat *website* sistem rekomendasi tempat makan, sehingga dapat memuaskan pengalaman *user* (*user experience*) dalam mencari informasi dan rekomendasi tentang makanan apa saja yang terdapat di suatu daerah yang akan atau sedang dikunjunginya.

Sistem rekomendasi tersebut menggunakan prefensi penggunanya sendiri sebagai acuan. Sistem akan mencatat jejak pencarian dan pemilihan makanan-makanan yang dicari, kemudian memberikan rekomendasi tempat makan yang memiliki kesamaan dengan yang dipilih oleh *user*, atau bahkan lebih baik dari perkiraan *user* sendiri.

Dari pemaparan di atas, maka dirancanglah sistem rekomendasi tempat makan berbasis *web* dengan algoritma *Squeezer* yang dapat membantu memberikan rekomendasi tempat makan terhadap pengguna *web* tersebut berdasarkan kriteria dari makanan yang dicari atau sedang dilihat. Sistem ini juga menggunakan metode *Term Frequency Ranking* yang dapat membantu memberikan rekomendasi tempat makan berdasarkan kata pencarian (*keyword*) dari *user* dengan memberikan rekomendasi tempat makan yang memiliki keterkaitan dan diurutkan berdasarkan hasil yang memiliki keterkaitan paling tinggi ke paling rendah.

Penelitian ini merupakan pengembangan dari penelitian tentang algoritma *Squeezer* sebelumnya, yang berjudul "Penerapan Algoritma *Squeezer* untuk Memberikan Rekomendasi Pilihan Lagu Berdasarkan Daftar Lagu yang Dimainkan pada Pemutar Mp3 Android" oleh Eko Wahyu Wibowo, Siti Rochimah, dan Abdul Munif. Pada penelitian sebelumnya, algoritma *Squeezer* diuji coba pada kumpulan data yang sudah ada. Pada penelitian ini, algoritma *Squeezer* digunakan untuk membaca atau merekam perilaku dan input dari *user*. Sampai saat ini, belum ditemukan penelitian serupa tentang sistem rekomendasi tempat

makan. Diharapkan penelitian ini dapat menjadi acuan dan materi yang berguna bagi penelitian selanjutnya.

II. LANDASAN TEORI

A. Algoritma *Squeezer*

Squeezer merupakan sebuah algoritma yang digunakan untuk mengelompokkan data (*clustering*) sekumpulan data bertipe kategorikal [1]. 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 ke dalam 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.

Berikut adalah definisi yang digunakan dalam algoritma *Squeezer*. Misal A1, ..., Am adalah himpunan kategorikal atribut dengan domain D1, ..., Dm berturut-turut. Misal himpunan data D adalah himpunan dari pasangan data dimana setiap pasangan data t:t∈D"1 x … x Dm ". Misal TID adalah himpunan dari ID unik dari setiap pasangan data. Untuk setiap tid ∈ TID, nilai atribut A1 dari pasangan data yang bersangkutan direpresentasikan sebagai val(tid, Ai).

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*

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*()

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()*

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()*

B. *Ranking Model*

Untuk menetapkan peringkat dari sebuah dokumen, dilakukan pemberian skor terhadap setiap dokumen sebagai estimasi relevansi dokumen dengan query atau kata pencari yang diberikan [2]. Pembobotan dan ranking dapat diuji nilai efektifitasnya dengan mengukur nilai *Precision* dan *Recall*. *Precision* adalah jumlah data yang relevan yang ditemukan dibagi jumlah data yang ditemukan. Sementara *Recall* adalah jumlah data relevan yang ditemukan dibagi jumlah data relevan. Suatu algoritma atau metode memiliki relevansi tinggi apabila memiliki nilai *precision* dan *recall* mendekati 1 [2].

Pembobotan dalam sebuah dokumen dapat ditentukan dan dilakukan dengan tiga cara [3], yaitu :

1. Term Frequency, merupakan jumlah kemunculan suatu kata atau istilah pada dokumen atau data terkait. Secara intuitif, semakin banyak kata atau istilah yang cocok atau sesuai dengan kata pencari, maka semakin tinggi bobot dokumen tersebut yang juga mengacu semakin tinggi nilai relevansi dokumen atau data tersebut dengan kata pencari.

2. Document Frequency, merupakan jumlah banyaknya dokumen atau data dimana istilah tersebut muncul pada sebuah kumpulan data. Secara intuitif, semakin banyak data yang memiliki istilah tersebut, semakin buruk istilah tersebut sebagai diskriminator. Oleh karena itu, bobot yang diberikan pada istilah tersebut sangat kecil.

3. Document Length, merupakan istilah panjang dokumen atau data yang dapat berupa *size* dalam *byte* atau jumlah banyaknya istilah yang dikandung oleh dokumen atau data tersebut. Semakin panjang sebuah dokumen, maka istilah tersebut akan semakin sering muncul dan digunakan (memiliki frekuensi tinggi), sehingga dapat memberikan bobot yang lebih tinggi.

C. Sistem Rekomendasi

Rekomendasi adalah memberitahukan kepada seseorang atau lebih bahwa sesuatu yang dapat

dipercaya. Dapat juga merekomendasikan diartikan sebagai menyarankan, mengajak untuk bergabung, menganjurkan suatu bentuk perintah. Sistem rekomendasi bertujuan memberikan rekomendasi *item* (barang-barang) baru kepada *user* (pengguna/calon pembeli) yang akan digunakan. Sistem rekomendasi biasanya menghasilkan daftar rekomendasi di salah satu dari dua cara yaitu melalui penyaringan kolaboratif atau berbasis konten [4]. Pendekatan kolaboratif membangun sebuah model dan perilaku masa lalu pengguna (data yang sebelumnya dipilih oleh pengguna), kemudian menggunakan data tersebut untuk memprediksi data baru yang nantinya akan direkomendasikan kepada pengguna.

Terdapat dua pendekatan utama untuk merekomendasikan *item* kepada *user*, yaitu *content-based filtering* dan *collaborative filtering* [5].

1. Content-Based Filtering

Content-Based Filtering menggunakan informasi-informasi yang sudah ada sebelumnya untuk diperhitungkan kembali untuk menjadi *output* dalam memberikan rekomendasi. Pendekatan ini dapat memberikan rekomendasi, walaupun belum ada timbal balik atau input terlebih dahulu dari *user*.

2. Collaborative Filtering

Collaborative Filtering menggunakan timbal balik dari *user* terhadap sistem untuk meningkatkan kualitas materi yang diberikan kepada *user*. Untuk dapat memberikan rekomendasi terhadap *user*, *user* harus terlebih dahulu memberikan *input* atau masukkan terhadap sistem. *Input* dari *user* tersebut akan dicatat oleh sistem dan diolah menjadi data yang nantinya akan diberikan kepada *user*.

Untuk sistem rekomendasi tempat makan daerah ini, digunakan *Collaborative-Filtering* yang mengambil timbal balik dari *user* terhadap *item* yang dipilih. Kemudian dengan algoritma *Squeezer* diambil kriteria dari *item* yang dimasukkan *user* dan diproses.

III. METODOLOGI DAN PERANCANGAN SISTEM

A. Metodologi Penelitian

1. Studi Literatur

Tahap pertama penelitian ini adalah dengan melakukan studi literatur menggunakan referensi buku, artikel, jurnal, mengenai algoritma *Squeezer*, metode *Term Frequency ranking*, dan berbagai sumber untuk mendukung pembangunan dan pengembangan *website*. Literatur tersebut dijadikan pedoman dalam melakukan penelitian.

2. Pencarian dan Pendataan Kebutuhan

Pengumpulan data-data berupa tempat makan di kota-kota yang sudah ditentukan sebelumnya. Data-data tempat makan tersebut diberikan beberapa atribut yang merupakan kriteria makanan tersebut.

3. Perancangan Desain *Website* dan *Database*

Setelah mendapatkan semua data yang dibutuhkan, dilakukan perancangan dan pembuatan *database* untuk menampung semua data yang telah didapatkan. Selain itu, dilakukan juga perancangan *website* yang nantinya akan digunakan untuk mempresentasikan data.

4. Pembangunan *Website* dan Penerapan Algoritma

Melakukan pembangunan *website* dan implementasi algoritma *Squeezer* yang digunakan dalam sistem rekomendasi ini.

5. Uji Coba

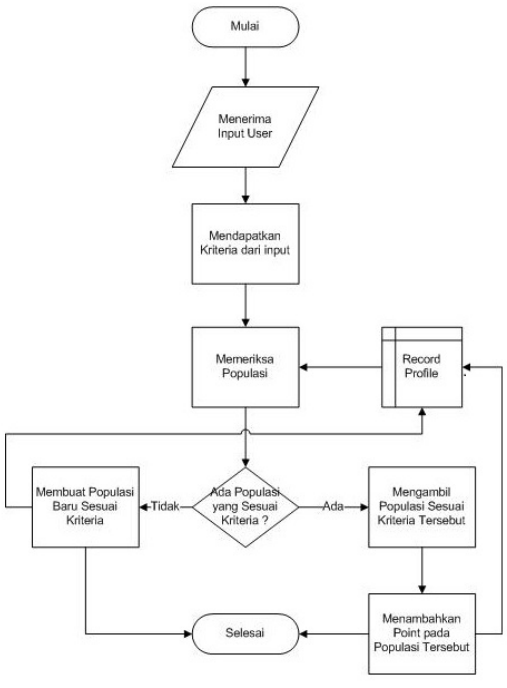
Website yang telah jadi dan diimplementasikan dengan Algoritma *Squeezer* untuk sistem rekomendasinya, akan dicoba seberapa akuratnya rekomendasi yang dihasilkan oleh Algoritma *Squeezer* ini. Pengujian ini menggunakan beberapa skenario yang telah disusun sesuai dengan kemungkinan – kemungkinan yang akan terjadi untuk melihat hasil yang dikeluarkan oleh algoritma *Squeezer* ini dalam pemberian rekomendasi kepada *user*. Pengujian terhadap metode Te*rm Frequency Ranki*ng dilakukan dengan memasukkan *string* kata pencarian dengan panjang yang bervariasi. Keakuratan dari metode *Term Frequency Ranking* ini dapat diukur dari nilai *Precision* dan *Recall* yang akan keluar sesuai dengan string kata pencari yang dimasukkan.

B. Perancangan Sistem

1. Diagram alir

- Diagram Alir Sub-proses *Squeezer*

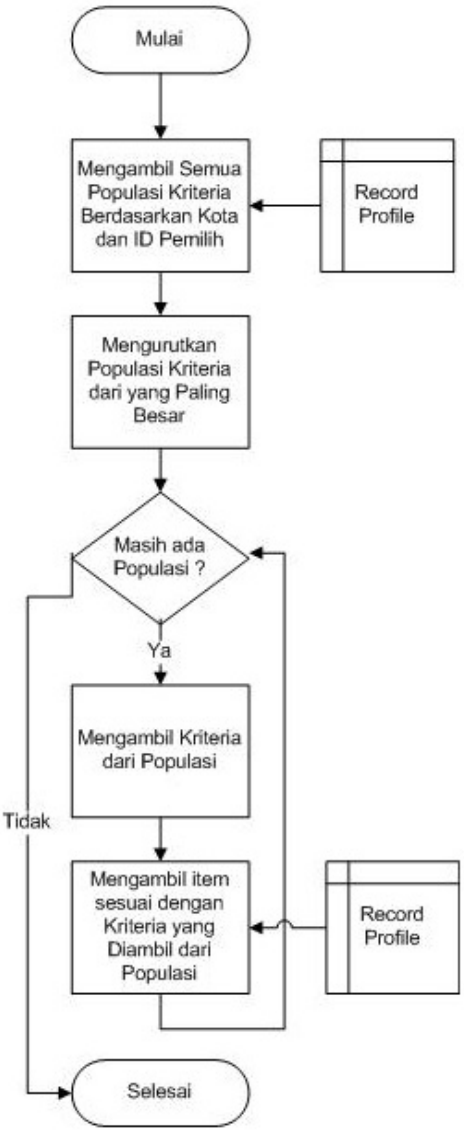
➔ Diagram alir pembentukan populasi



Gambar 5. Diagram alir pembentukan populasi

Pada sub-proses ini, dimulai dari penerimaan input dari user, yaitu pilihan tempat makan. Pilihan tempat makan tersebut akan diambil kriteria yang terdapat didalamnya. Setelah itu, akan dilakukan pemeriksaan apakah terdapat populasi yang sesuai dengan kriteria tersebut. Apabila tidak terdapat populasi yang sesuai dengan kriteria tersebut, maka akan dibuatkan populasi baru yang sesuai dengan kriteria tersebut. Apabila sudah ada populasi yang sesuai dengan kriteria tersebut, maka akan ditambahkan point pada populasi tersebut.

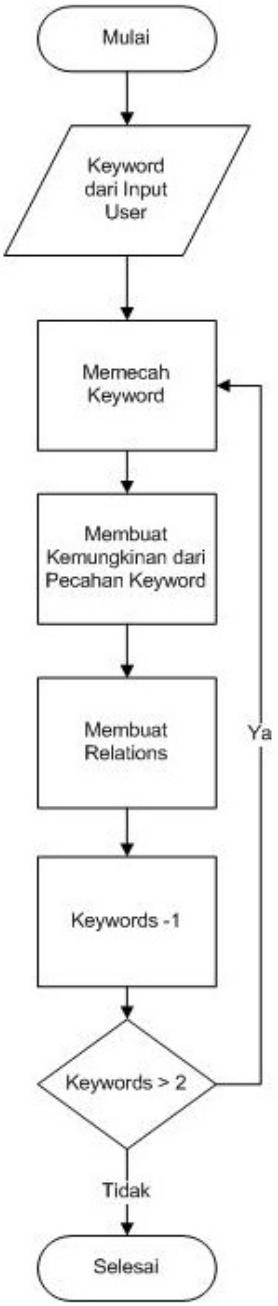
➔ Diagram alir pemberian rekomendasi



Gambar 6. Diagram alir pemberian rekomendasi

Pada bagian pemberian rekomendasi ini, dimulai dari pengambilan semua populasi kriteria yang sesuai dengan user yang sedang aktif dan kota tempat makan yang dipilih. Populasi-populasi tersebut diurutkan dari yang memiliki *point* paling tinggi sampai ke *point* yang paling rendah. Setelah itu, kriteria-kriteria pada setiap populasi akan diambil dan dijadikan acuan untuk memberikan rekomendasi kepada *user*, yaitu dengan cara merekomendasikan tempat makan yang memiliki kriteria yang sama dengan kriteria pada populasi tersebut.

- Diagram Alir Sub-proses *Ranking*



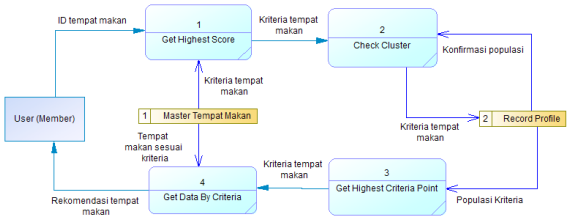
Gambar 7. Diagram alir sub-proses *Ranking*

Pemberian *ranking* untuk menentukan relevansi dokumen dimulai dengan penerimaan kalimat pencari dari *user*. Kalimat pencari tersebut kemudian dipecah

menjadi kata-kata dan kemudian dilakukan dan dicari kemungkinan-kemungkinan kombinasi kata yang terbentuk. Setelah semua kemungkinan kombinasi kata terbentuk, maka jumlah panjang kata untuk kemungkinan kombinasi akan dikurangi dan dilakukan pencarian kembali kombinasinya. Data yang memiliki kata pencari yang lebih banyak didalamnya, memiliki nilai relevansi lebih tinggi. Setelah itu, dibuatlah pengurutan dari data yang memiliki relevansi paling tinggi ke yang paling rendah.

2. *Data Flow Diagram*

- DFD Algoritma *Squeezer*

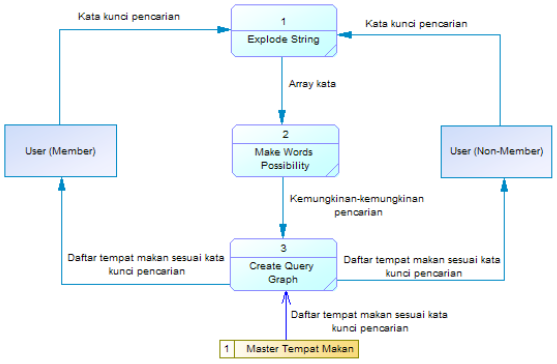


Gambar 8 . DFD Algoritma *Squeezer*

Pada DFD *Squeezer* seperti pada Gambar 8, terdapat empat proses utama pemberian rekomendasi, yaitu *Get Highest Score*, *Check Cluster*, Get *Highest Criteria Point,* dan *Get Data by Criteria*. Proses dimulai dari *user* yang memilih sebuah tempat makan, secara otomatis *ID* tempat makan tersebut dikirimkan. Kemudian, pilihan tempat makan oleh *user* tersebut diolah untuk diambil nilai kriteria tertingginya pada proses *Get Highest Score*. Setelah nilai kriteria tertingginya diperoleh, dilakukan pemeriksaan pada populasi apakah sudah ada populasi dengan kriteria yang didapat tersebut atau belum. Apabila belum ada, maka dibuat populasi baru, dan apabila sudah ada, maka ditambahkan 1 point pada populasi tersebut.

Proses selanjutnya adalah *Get Highest Criteria Point* dimana pemeriksaan terhadap populasi yang ada. Populasi-populasi tersebut diambil dan diurutkan berdasarkan *point* yang paling tinggi ke yang paling rendah. Setelah itu, dilakukan pengambilan data-data yang sesuai dengan kriteria dari populasi-populasi yang diambil untuk diberikan kepada *user* pada proses *Get Data by Criteria*.

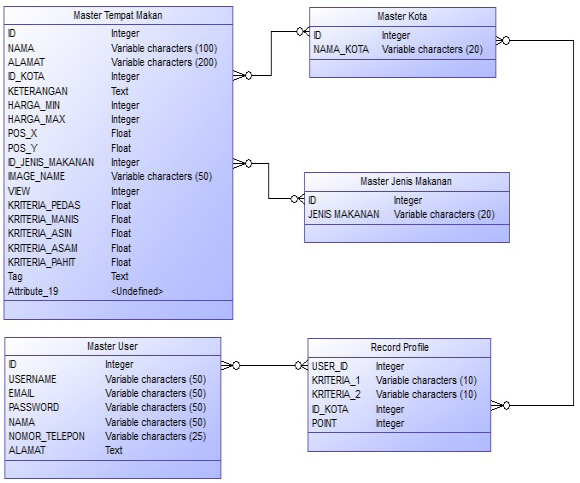
- DFD *Term Frequency Ranking*



Gambar 9. DFD *Term Frenquency Ranking*

Proses ranking ini digunakan pada saat *user* memberikan kata kunci pencarian untuk mencari tempat makan yang diinginkan. Dimulai pada proses *Explode String*, dimana kata kunci oleh *user* dipecah menjadi pecahan-pecahan kata. Kemudian, pecahan-pecahan kata tersebut dikirimkan ke proses *Make Words Possibility* untuk dibuatkan kemungkinan-kemungkinan kombinasi kata. Setelah semua kemungkinan kombinasi kata sudah terbentuk, maka kata-kata tersebut dijadikan *query* untuk melakukan penarikan dari database pada proses *Create Query Graph*. Kemudian hasil *query* tersebut diberikan kepada *user* pada halaman *Search Result*.

3. *Entity Relationship Diagram*



Gambar 10. *Entity Relationship Diagram*

C. Penerapan

1. Algoritma *Squeezer*

Gambar 11. Kode Sumber Algoritma *Squeezer*

2. Metode *Term Frequency Ranking*

```
function start()
{
    $this->db = mysql_connect($this->host, $this->user, $this->password);
    mysql_select_db($this->database);


    $this->explode_string();
    $this->make_graph();
    $this->check();
}

function explode_string()
{
    // memecahkan kalimat pencari menjadi kata-kata
    $this->strings = explode(" ", $this->keywords);

    // menghitung jumlah kata yang ada
    foreach ($this->strings as $string)
    {
        $this->words++;
    }
}

function make_graph()
{...23 lines }

function check()
{
    $flag = 0 ;

    foreach ($this->string_arr as $keyword)
    {
        $search = explode(" ", $keyword);
        if ($flag == 0)
        {
            $query = "SELECT * from master_tempat_makan WHERE ";
            $flag = 1;
        }
        else
        {
            $query .=" UNION ALL SELECT * from master_tempat_makan WHERE ";
        }

        $i=0;

        foreach ($search as $search_explode)
        {
            if ($i==0)
                {
                    $query .="tag LIKE '%$search_explode%'";
                }
            else
                {
                    $query .="AND tag LIKE '%$search_explode%'";
                }
            $i++;
        }
    }

    @$this->result = mysql_query($query);
}
```

Gambar 12. Kode Sumber Metode *Term Frequency Ranking*

IV. UJI COBA

A. Uji Coba Algoritma *Squeezer* dalam Pemberian Rekomendasi

Pengujian ini dilakukan dengan membuat empat skenario pemilihan makanan dengan perbandingan makanan yang memiliki dua kriteria dengan makanan yang memiliki satu kriteria yang berbeda.

Berikut adalah skenario-skenario pemilihan makanan dan hasil yang dikeluarkan oleh algoritma *Squeezer* dalam pemberian rekomendasi :

1) 1 kriteria = 0 , 2 kriteria = 15

Percobaan	Pilihan		Populasi (User)		Populasi (Sistem)		Kesimpulan
			Kriteria	Point	Kriteria	Point	
15	pedas - manis	pedas - manis	pedas - manis	5	pedas	13	Tidak Sesuai
	pedas - asin	manis - asin	pedas - asin	3	manis	7	
	pedas - asin	pedas - pahit	pedas - asam	3	pedas - manis	5	
	manis - asin	pedas - asam	pedas - pahit	2	asin	5	
	pedas - manis	pedas - manis	manis - asin	2	pedas - asin	3	
	pedas - asin	pedas - manis			pedas - asam	3	
	pedas - asam	pedas - pahit			pedas - pahit	2	
	pedas - asam				manis - asin	2	
					asam	3	
					pahit	2	

2) 1 kriteria = 5 , 2 kriteria = 10

Percobaan	Pilihan		Populasi (User)		Populasi (Sistem)		Kesimpulan
			Kriteria	Point	Kriteria	Point	
15	pedas -manis	manis	pedas - manis	4	pedas	8	kurang sesuai
	pedas - asin	asin	pedas - asin	2	manis	7	
	pedas - manis	asin	manis - asin	2	asin	5	
	pedas - asam	asam	manis	2	pedas - manis	4	
	manis - asin	manis	asin	2	pedas - asin	2	
	manis - asin		pedas - asam	1	manis - asin	2	
	pedas - manis		pedas - pahit	1	asam	2	
	pedas - manis		asam	1	pedas - asam	1	
	pedas - asin				pedas - pahit	1	
	pedas - pahit				pahit	1	

3) 1 kriteria = 10 , 2 kriteria = 5

Percobaan	Pilihan		Populasi (User)		Populasi (Sistem)		Kesimpulan
			Kriteria	Point	Kriteria	Point	
15	manis	pedas - manis	manis	4	manis	7	Cukup sesuai
	manis	pedas - asin	pedas	3	pedas	7	
	pedas	pedas - manis	asin	2	asin	4	
	manis	pedas - asam	pedas - manis	2	pedas - manis	2	
	asin	manis - asin	pedas - asin	1	asam	2	
	pedas		pedas - asam	1	pedas - asin	1	
	asin		manis - asin	1	pedas - asam	1	
	manis		asam	1	manis - asin	1	
	pedas						
	asam						

4) 1 kriteria = 15 , 2 kriteria = 0

Percobaan	Pilihan		Populasi (User)		Populasi (Sistem)		Kesimpulan
			Kriteria	Point	Kriteria	Point	
15	manis	manis	manis	6	manis	6	Sesuai
	asin	asin	asin	4	asin	4	
	asam	manis	pedas	3	pedas	3	
	manis	asam	asam	2	asam	2	
	pedas	manis					
	asin	pedas					
	manis	pedas					
	asin						

Pada percobaan pertama (Pada tabel 1), dilakukan pemilihan 15 makanan yang semuanya memiliki dua kriteria. Hasil populasi yang diberikan oleh sistem tidak sesuai dengan populasi pilihan *user* yang sebenarnya. Hal ini terjadi karena pada saat pemilihan makanan yang memiliki dua kriteria, akan dibuatkan atau ditambahkan *point* pada kriteria yang menjadi bagiannya (penambahan *point* pada tiga populasi sekaligus). Sebagai contoh, untuk makanan yang memiliki kriteria pedas – manis, maka akan diberikan *point* pada populasi yang memiliki kriteria pedas, manis, dan pedas – manis. Hal ini dapat menyebabkan kriteria tunggal memiliki nilai tinggi karena gabungan dari makanan yang memiliki dua kriteria lainnya yang dipilih.

Sementara percobaan kedua (pada tabel 2), dilakukan pemilihan lima makanan yang memiliki satu kriteria dan 10 makanan yang memiliki dua kriteria. Hasil populasi yang dikeluarkan oleh sistem kurang sesuai dengan populasi pilihan *user* yang sebenarnya walaupun masih lebih baik daripada percobaan dengan skenario pemilihan 15 makanan yang memiliki dua kriteria (percobaan pertama) tadi. Hal ini karena jumlah makanan yang memiliki dua kriteria yang dipilih lebih sedikit daripada pada percobaan pertama. Hasil bias yang dikeluarkan dapat diperkecil.

Pada percobaan ketiga (pada tabel 3), dilakukan pemilihan 10 makanan yang memiliki satu kriteria dan lima makanan yang memiliki dua kriteria. Hasil populasi yang dikeluarkan oleh sistem cukup sesuai dengan populasi pilihan *user* yang sebenarnya.

Terdapat tiga kriteria teratas yang sama antara populasi yang dikeluarkan oleh sistem dengan populasi pilihan *user* yang sebenarnya.

Sementara pada percobaan terakhir ini (pada tabel 4), dilakukan pemilihan 15 makanan yang memiliki satu kriteria. Hasil populasi yang dikeluarkan oleh sistem sesuai dengan populasi pilihan *user* yang sebenarnya. Hal ini dikarenakan tidak terdapat pemilihan makanan yang memiliki dua kriteria yang menyebabkan hasil yang dikeluarkan oleh sistem menjadi kurang akurat.

Dari keempat percobaan yang telah dilakukan, dapat disimpulkan bahwa jumlah kriteria makanan yang dipilih oleh *user* sangat mempengaruhi populasi yang dihasilkan oleh algoritma *Squeezer* yang digunakan untuk pemberian rekomendasi kepada *user*. Dengan pemilihan terhadap makanan yang memiliki dua kriteria, akan memberikan point kepada tiga populasi yang terkait sehingga dapat menyebabkan nilai populasi–populasi tersebut menjadi bias. Semakin banyak *user* memilih makanan dengan dua kriteria, maka rekomendasi yang diberikan akan semakin tidak sesuai.

Hal tersebut menunjukkan bahwa masih ada kelemahan terhadap algoritma *Squeezer* apabila hanya menggunakan kriteria makanan sebagai input dalam pembentukan populasi dan mengolahnya menjadi rekomendasi. Untuk mengatasi kelemahan tersebut, diperlukan penelitian lebih lanjut dalam penggunaan variabel – variabel lainnya seperti jenis makanan, maupun relasi antar makanan yang juga dapat digunakan sebagai acuan untuk membentuk populasi, tidak hanya populasi menurut kriteria makanan saja. Dengan pembentukan relasi antar populasi tersebut, diharapkan keakuratan dan kesesuaian rekomendasi yang dikeluarkan oleh algoritma *Squeezer* ini dapat menjadi lebih tinggi.

B. Uji Coba Performa Metode *Term Frequency Ranking*

Pengujian ini dilakukan dengan menggunakan *String* acak ke dalam sistem. *String* tersebut antara lain:

1. “angkringan”
2. “soto ayam”
3. “soto ayam jogja”
4. “nasi rames jogja”
5. “angkringan malam jogja”
6. “nasi gandul pak subur”

Berikut adalah hasil dari percobaan yang dilakukan.

Tabel 5. Tabel Hasil Pengujian metode *Term Frequency Ranking*

Percobaan ke	String	Relevan Item	Retrieved Item	Relevan Collection	Precision	Recall
1	1	2	2	2	1	1
2	2	3	3	3	1	1
3	3	2	8	2	0.25	1
4	4	1	7	1	0.14	1
5	5	1	4	1	0.25	1
6	6	1	8	1	0.125	1
Rata-rata					**0.4608333**	**1**

Pada pengujian relevansi jawaban, metode *Term Frequency Ranking* ini memiliki rata-rata nilai *precision* dan *recall* sebesar 0.46 dan 1. Pada implementasi ini, nilai yang dihasilkan tidak terlalu tinggi, dikarenakan pengambilan data diambil secara acak dan hanya memperhitungkan kata-kata yang terdapat di dalamnya. Kelemahan metode *Term Frequency Ranking* ini adalah hanya menampilkan hasil berdasarkan perhitungan urutan kata saja. Apabila terdapat hasil yang tidak sesuai namun memiliki urutan kata benar, maka hasil tersebut akan ditampilkan sebagai rekomendasi. Untuk mengatasi kelemahan ini, diperlukan penelitian terhadap metode lainnya untuk mem-*filter* output yang dihasilkan, salah satunya dengan pemeriksaan terhadap frase dari kalimat tersebut apakah benar atau tidak. Diharapkan dengan penggunaan filter tersebut dapat meningkatkan akurasi dan ketepatan rekomendasi yang dihasilkan dari metode *Term Frequency Ranking* ini. Namun kelebihan dari metode *Term Frequency Ranking* ini adalah mengurutkan item dari yang memiliki relevansi paling tinggi sampai paling rendah.

V. SIMPULAN DAN SARAN

A. Simpulan

Algoritma *Squeezer* telah berhasil diimplementasikan di dalam sistem rekomendasi tempat makan. Dari penelitian yang telah dilakukan, dapat disimpulkan bahwa rekomendasi yang dihasilkan oleh algoritma *Squeezer* ini sangat bergantung pada pilihan *user*. Semakin banyak *user* memilih makanan yang memiliki dua kriteria, maka rekomendasi yang diberikan semakin tidak sesuai. Hal tersebut dikarenakan pemilihan makanan yang memiliki dua kriteria akan memberikan nilai kepada tiga populasi sekaligus dan membuat perhitungan menjadi tidak tepat.

Relevansi jawaban yang diberikan dalam metode *Term Frequency Ranking* ini memberikan hasil yang cukup baik. Berdasarkan pengujian, metode ini memiliki tingkat *recall* yang tidak terlalu tinggi yaitu 1, dan *precision* 0.46 dimana semua *item* yang memiliki relevansi ditampilkan kepada *user*. Akan tetapi kelemahan metode ini adalah hanya menampilkan hasil yang memiliki kesesuaian dengan kata saja dan tidak melihat kesesuaian *item* yang direkomendasikan tersebut. Selain itu, kelebihan penggunaan metode ini

yaitu menampilkan *item* dengan urutan dari *item* yang memiliki relevansi paling tinggi sampai paling rendah

B. Saran

Beberapa saran yang dapat diberikan untuk mengembangkan penelitian berikutnya antara lain:

1. Agar pemberian rekomendasi oleh algoritma *Squeezer* ini dapat lebih akurat, dapat menggunakan variabel – variabel lain seperti jenis makanan, ataupun relasi antar makanan dalam pembentukan populasi. Selain itu, perlu juga dilakukan penghubungan antar populasi dengan variabel yang berbeda dan dilakukan validasi, sehingga rekomendasi yang diberikan akan lebih akurat dan tidak mengacu pada satu variabel saja.
2. Untuk menghasilkan hasil pencarian yang lebih baik dan optimal dalam hal relevansi, disarankan menambahkan fitur *filtering* terhadap kalimat dari item yang direkomendasikan. Filter tersebut dapat berupa pemeriksaan frase dari kalimat, pembuangan kata yang tidak baku, dan sebagainya, sehingga dapat menampilkan hasil yang lebih sesuai dan akurat sehingga dapat meningkatkan nilai *precision* dan *recall* nya.

Daftar Pustaka

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

[2] Wibowo, Ari. 2011. "Pengujian Kerelevanan Sistem temu Kembali Informasi". Seminar Nasional Ilmu Komputer 2011.

[3] Liu, F. dkk, "Effective Keyword Search in Relational Database". Proceeding ACM SIGMOD International Conference on Management of Data. Pp. 563-574.

[4] F. Ricci, L. Rokach, B. Shapira. Chapter 1: Introduction to Recommender Systems Handbook. Recommender Systems Handbook. Springer. 2011.

[5] Hosein Jafarkarimi, "A Naïve Recommendation Model for Large Databases". International Journal of Information and Education Technology. 2012.

[6] E.W Wibowo, Siti R., A. Munif.2013. "Penerapan Algoritma Squeezer untuk Memberikan Rekomendasi Pilihan Lagu Berdasarkan Daftar Lagu yang Dimainkan pada Pemutar Mp3 Android". Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS), Vol. 2 No. 1, 2013.