Journal of Informatics and Telecommunication Engineering

Available online <http://ojs.uma.ac.id/index.php/jite>

Implementasi Algoritma Apriori Pada Data Benchmark Kosarak Dan Mushrooms

Implementation of Apriori Algorithm on Data of Kosarak and Mushrooms Benchmark

Rizki Muliono*

Program Studi Teknik Informatika, Fakultas Teknik
Universitas Medan Area, Indonesia

*Corresponding author: E-mail : rizkimuliono@gmail.com

Abstrak

Algoritma apriori saat ini lebih banyak digunakan untuk mencari frequent itemsets dan mencari aturan asosiasi untuk menemukan knowledge. Proses mencari frequent itemsets pada data secara berulang-ulang yang ada didalam database dan diakhiri ketika kandidat itemsets sampai $K+1$ tidak ada lagi. Algoritma Apriori menggunakan secara umum menggunakan banyak jumlah memori dan waktu eksekusi dalam menemukan kombinasi dan perbandingan frequent itemsets. Hasil yang di dapatkan dengan menggunakan algoritma apriori bisa di katakan akurat saat menseleksi kombinasi itemset yang ada pada dataset sesuai dengan nilai support dan confidens nya. Untuk mengetahui seberapa akurat dan berapa jumlah sumberdaya yang di gunakan serta bagaimana perilaku algoritma apriori terhadap dataset dengan jumlah kolom data yang berbeda, maka implementasi algoritma apriori di ujikan dengan data benchmark kosarak.dat dan mushrooms.dat dengan nilai minimum support yang sama. Kedua data sets tersebut memiliki format yang berbeda pada jumlah kolom datanya yaitu data pada semua baris memiliki jumlah kolom karakter data, pada datasets kosarak.dat memiliki kolom karakter dengan panjang berbeda-beda pada setiap barisnya sedangkan pada datasets mushrooms.dat memiliki kolom karakter sebanyak 23 karakter data, artinya datasets tersebut memiliki model blok data linear atau sama. Hasil dari implementasi algoritma apriori terhadap kedua datasets tersebut didapatkan perilaku proses pada apriori yang ditampilkan dari hasil waktu eksekusi dan memori yang dipakai bahwa datasets kosarak lebih sedikit menggunakan waktu dibandingkan dengan datasets mushrooms namun penggunaan memori lebih boros, semakin kecil nilai minimum support semakin banyak komparasi kandidat yang dicari.

Kata Kunci : apriori; *datamining*; implementasi; kosarak; mushrooms

Abstract

Apriori algorithm is currently used for searching frequent itemsets and looking for association rules to find knowledge. The process of finding frequent itemsets on the data repeatedly in the database and ended when candidate itemsets until $K + 1$ no longer exists. Algoritma Apriori uses in general use a large amount of memory and execution time in finding combinations and frequent itemsets comparison. The results obtained by using a priori algorithm can be said accurately when selecting a combination itemset on the dataset in accordance with the value of support and confidens it. To find out how accurate and how many resources are in use and how the a priori algorithm's behavior to the dataset with different data columns, the a priori agri- prim implementation is tested with benchmark data of kosarak.dat and mushrooms.dat with the same minimum support value. Both of these sets of data have different formats on the number of columns of data that is data on all lines have the number of columns of data characters, the character column kosarak.dat memiliki datasets with different lengths on each line while the datasets mushrooms.dat have as many as 23 characters with character columns Data, meaning the datasets have a linear or similar data block model. The results of the algorithm implementation priori to both datasets is obtained behavior of processes on a priori appears from the results of the execution time and memory use that datasets kosarak less use of time compared with datasets mushrooms but use more memory intensive, smaller value of minimum support each comparison candidate Searched.

Keywords : Apriori; *Datamining*; implementation, kosarak, mushrooms

How to Cite: Muliono, R. 2017, Implementasi Algoritma Apriori Pada Data Benchmark Kosarak Dan Mushrooms, *Journal of Informatics and Telecommunication Engineering*, 1(1) :34-41.

PENDAHULUAN

Algoritma Apriori adalah salah satu algoritma data mining yang digunakan untuk mencari frequent itemsets dari database. Pencarian itemsets pada apriori harus memiliki parameter: minimum support dan minimum confidence[3], Pola pencarian kombinasi frequent itemsets pada apriori ini menggunakan metode pencarian breadth first search[7]. Tujuan algoritma apriori menemukan semua frequent itemset dengan mengulangi proses diatas dengan mencari secara berulang-ulang semua yang ada di dalam dataset dan diakhiri ketika kandidat itemsets dihasilkan C_{k-1} [1]. dan proses perulangan apriori tersebut membuat algoritma ini kurang efisien[6], Proses tersebut adalah join dan pemangkasan atau pruning[4].

Algoritma apriori bekerja dengan cara menghasilkan kandidat baru dari k -itemset pada frequent itemsets sebelumnya dan menghitung nilai support k -itemset tersebut. Itemsets yang memiliki nilai support di bawah dari *minsup* akan dihapus. Algoritma apriori memiliki Pseudocode di bawah ini oleh Agrawal & Srikant (1993).

```

 $L_1 := \{ \text{large 1-itemsets} \};$ 
 $k := 2;$  // k represents the pass number
while ( $L_{k-1} \neq \emptyset$ ) do
begin
   $C_k :=$  New candidates of size  $k$  generated from  $L_{k-1}$ ; (apriori-gen)
  forall transactions  $t \in D$  do
    Increment the count of all candidates in  $C_k$  that are contained in  $t$ ;
   $L_k :=$  All candidates in  $C_k$  with minimum support;
   $k := k + 1$ ;
end
Answer :=  $\bigcup_k L_k$ ;

```

Gambar 1. Pseudocode Algoritma Apriori, Agrawal & Srikant (1993).

Algoritma Apriori menggunakan pengetahuan frekuensi atribut yang telah diketahui sebelumnya untuk memproses informasi selanjutnya. Pada algoritma Apriori menentukan kandidat yang mungkin muncul dengan cara memperhatikan minimum support dan minimum confidence. Support adalah nilai pengujung atau persentase kombinasi sebuah item dalam database.

Transaction D		C_1		L_1	
TID	Items	Itemset	Count	Itemset	Count
100	1,3,4	{1}	2	{1}	2
200	2,3,5	{2}	3	{2}	3
300	1,2,3,5	{3}	3	{3}	3
400	2,5	{4}	1	{5}	3
		{5}	3		

C_2		C_2		L_2	
Itemset		Itemset	Count	Itemset	Count
{1,2}		{1,2}	1	{1,3}	2
{1,3}		{1,3}	2	{2,3}	2
{1,5}		{1,5}	1	{2,5}	3
{2,3}		{2,3}	2	{3,5}	2
{2,5}		{2,5}	3		
{3,5}		{3,5}	2		

C_3		C_3		L_3	
Itemset		Itemset	Count	Itemset	Count
{2,3,5}		{2,3,5}	2	{2,3,5}	2

Gambar 2. Ilustrasi Algoritma Apriori (Han et al, 2012).

frequent 1-itemset ditemukan dengan membaca database untuk mengumpulkan hitungan untuk setiap item, dan mengumpulkan item-item yang memenuhi minimum support. Set yang dihasilkan dilambangkan dengan L_1 . Selanjutnya, L_1 digunakan untuk menemukan L_2 , *frequent itemset 2-itemset*, yang digunakan untuk menemukan L_3 , dan seterusnya, sampai tidak ada lagi *frequent k-itemsets* dapat ditemukan. Pencarian setiap L_k dilakukan dengan proses pembacaan ulang pada database[1,2].

Setiap himpunan bagian itemsets yang suportnya lebih tinggi dari nilai minimum yang ditetapkan disebut *minimum support* dan seluruhnya disebut *frequent itemsets*[2]. *Frequent itemset* adalah sekumpulan item yang sering muncul secara bersamaan.

Dalam *association rule mining* terdiri dari dua sub persoalan :

1. Menemukan semua kombinasi dari item, disebut dengan *frequent itemsets*, yang memiliki support yang lebih besar dari pada *minimum support*.
2. Gunakan *frequent itemsets* untuk men-generate aturan yang dikehendaki. Semisal, ABCD dan AB adalah *frequent*, maka didapatkan aturan $AB \rightarrow CD$ jika rasio dari $support(ABCD)$ terhadap $support(AB)$ sedikitnya sama dengan *minimum confidence*. Aturan ini memiliki minimum

support karena ABCD adalah *frequent*.

$$\text{Support (A)} = \left(\frac{\text{jumlah transaksi mengandung A}}{\text{Total transaksi}} \right) \times 100\%$$

$$\text{Confidence } P(B|A) = \frac{\text{Total transaksi mengandung A dan B}}{\text{Transaksi mengandung A}} \times 100\%$$

Association rule adalah salah satu teknik utama atau prosedur dalam *Market Basket Analysis* untuk mencari hubungan antar item dalam suatu *data set* dan menampilkan dalam bentuk *association rule*[8]. Association rule (aturan asosiatif) akan menemukan pola tertentu yang mengasosiasikan data yang satu dengan data yang lain. Kemudian untuk mencari *association rule* dari suatu kumpulan data, tahap hal pertama yang harus dilakukan adalah mencari *frequent itemset* terlebih dahulu. Setelah semua pola *frequent itemset* ditemukan, barulah mencari aturan asosiatif atau aturan keterkaitan yang memenuhi syarat yang telah ditentukan[9].

Benchmarking adalah pendekatan yang secara terus menerus mengukur dan membandingkan produk barang dan jasa, dan proses-proses dan praktik-praktiknya terhadap standar ketat yang ditetapkan oleh para pesaing atau mereka yang dianggap unggul dalam bidang tersebut. Dengan melakukan atau melalui *benchmarking*, suatu organisasi dapat mengetahui telah seberapa jauh mereka dibandingkan dengan yang terbaiknya[10]. Pada penelitian ini menggunakan data *benchmark* dari sumber dari *FIM Dataset Repository* yang dipromotori oleh *IBM Almaden Quest Research Group* bersumber dari <http://fimi.ua.ac.be/data/>. [5]

kosarak.dat adalah kumpulan *clickstream* dari portal berita di Hungaria, dengan tipe data *sparse* berupa angka dengan jumlah baris sebanyak 990.002 dan 41.270 item berbeda didalamnya. Dataset ini sudah banyak digunakan oleh

peneliti sebagai percobaan dalam mengembangkan ilmu data mining.

Tabel 1. Sampel isi data *kosarak.dat*

No	Isi data
1.	1 2 3
2.	1
3.	4 5 6 7
4.	1 8
5.	9 10
6.	11 6 12 13 14 15 16
7.	1 3 7
8.	17 18
9.	11 6 19 20 21 22 23 24
10.	1 25 3
11.	26 3
12.	11 27 6 3 28 7 29 30 31 32 33 34
13.	35 36 37
14.	6 2 38
15.	39 11 27 1 40 6 41 42 43 44 45 46
16.	47 3 48 7 49 50 51
17.	52 6 3 53
18.	54 1 6 55
...	11 6 56 57 58 59 60 61 62 63 64
...	...

Dataset *musrooms.dat* adalah dataset ini berasal dari pusat penelitian jamur beracun The Audubon Society Field panduan untuk North American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf. Jeff Schlimmer dan didonasikan ke publik pada 27 April 1987 sebagai dataset repository dalam ilmu data mining. Dataset ini memiliki spesifikasi dan deskripsi items nilai berupa kombinasi set angka sebanyak 8.124 baris dan 119 items, dataset tersebut memiliki 22 atribut[5].

Tabel 2. Deskripsi Atribut dataset *mushroom.dat*

Atribut	Domain information
1. cap-shape:	bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
2. cap-surface:	fibrous=f, grooves=g, scale=y, smooth=s
3. cap-color:	brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y

4. bruises?:	bruises=t,no=f
5. odor:	almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,pungent=p,spicy=s
6. gill-attachment:	attached=a,descending=d,free=f,notched=n
7. gill-spacing:	close=c,crowded=w,distant=d
8. gill-size:	broad=b,narrow=n
9. gill-color:	black=k,brown=n,buff=b,chocolate=h,gray=g,green=r,orange=o,pink=p,purple=u,red=e,white=w,yellow=y

Tabel 3. Sampel isi data *mushrooms.dat*

No	Isi data
1	1 3 9 13 23 25 34 36 38 40 52 54 59 63 67 76 85 86 90 93 98 107
2	113
3	2 3 9 14 23 26 34 36 39 40 52 55 59 63 67 76 85 86 90 93 99 108
4	114
5	2 4 9 15 23 27 34 36 39 41 52 55 59 63 67 76 85 86 90 93 99 108
6	115
7	1 3 10 15 23 25 34 36 38 41 52 54 59 63 67 76 85 86 90 93 98
8	107 113
9	2 3 9 16 24 28 34 37 39 40 53 54 59 63 67 76 85 86 90 94 99 109
10	114
...	2 3 10 14 23 26 34 36 39 41 52 55 59 63 67 76 85 86 90 93 98
...	108 114
...	2 4 9 15 23 26 34 36 39 42 52 55 59 63 67 76 85 86 90 93 98 108
...	115
...	2 4 10 15 23 27 34 36 39 41 52 55 59 63 67 76 85 86 90 93 99
...	107 115
...	1 3 10 15 23 25 34 36 38 43 52 54 59 63 67 76 85 86 90 93 98
...	110 114
...	2 4 9 14 23 26 34 36 39 42 52 55 59 63 67 76 85 86 90 93 98 107
...	115

METODE PENELITIAN

Dalam penelitian ini untuk mengetahui seberapa akurat dan berapa jumlah sumberdaya yang di gunakan serta bagaimana perilaku algoritma apriori terhadap dataset dengan jumlah kolom data yang berbeda, maka implementasi agoritma apriori di ujikan dengan data benchmark *kosarak.dat* dan *mushrooms.dat* dengan nilai minimum support yang sama. Kedua data akan di ujikan dengan nilai *minimum support* 0.010 -0.030 dan hasilnya akan di sajikan dalam bentuk tabel dan grafik analisis, yaitu tabel memori yang di pakai untuk proses apriori dan waktu yang di pakai yang di gunakan algoritma apriori terhadap dua data benchmark tersebut.

Di asumsikan sampel data tabel datanya adalah seperti berikut :

Tabel 4. Beberapa sampel dataset *kosarak.dat* sebagai data transaksi D

TID	Items
100	1,2,3,4
200	5,6,7,8,9
300	6,8,9,10
400	2,3,9

Langkah pertama adalah menginisialisasikann K=1 bangkitkan *C1-itemset* dan inialisasi item yang terkandung dalam transaksi dari tiap-tiap *k-item* tersebut sampai tidak ada anggota item yang tidak dikenali lagi dan menghitung berapa jumlah support yang terkandung dalam transaksi, kemudian baca item {1} kemudian telusuri tiap kolom dan baris transaksi jika menemukan item yang sama maka support item {1} ditambah 1. Proses yang sama berulang sampai seluruh item di-inialisasi berikut dengan nilai supportnya.

Tabel 5. Representasi Tabular data transaksi D

	TID			
items	100	200	300	400
{1}	1	0	0	0
{2}	1	0	0	1

{3}	1	0	0	1
{4}	1	0	0	0
{5}	0	1	0	0
{6}	0	1	1	0
{7}	1	0	0	0
{8}	0	1	1	0
{9}	0	1	0	1
{10}	0	0	1	0

Maka didapat himpunan items yang berbeda {1,2,3,4,5,6,7,8,9,10} adalah seluruh items yang terkandung di dalam data transaksi. Maka didapatkan untuk kandidat 1-itemset atau C1-itemsets seperti tabel 5 berikut ini.

Tabel 6. C1-Itemsets aprori data transaksi D

Items	Support	Items	Support
{1}	1	{6}	2
{2}	2	{7}	1
{3}	2	{8}	2
{4}	1	{9}	3
{5}	1	{10}	1

Bandingkan nilai support dari masing-masing item yang ada pada C1-itemset, jika nilai $min_sup = 2$ maka seluruh 1-itemset yang kurang dari 2 akan di eliminasi dari C1-itemset. Items yang dieliminasi adalah {1}:1,{4}:1,{5}:1,{7}:1,{10}:1 < 2. Hasil dari eliminasi maka didapat L1-itemset {2},{3},{6},{8},{9} dengan nilai supportnya dapat dilihat pada tabel 3.9 berikut :

Tabel 7. L1-Itemsets aprori data transaksi D

No	Items	Support Count
1	{2}	2
2	{3}	2
3	{6}	2
4	{8}	2
5	{9}	3

Selanjutnya adalah mengulangi langkah pertama dengan K+1, yaitu K=2. Mencari C2-itemset dari hasil L1-itemsets.

Kombinasi dari L1-itemset akan menghasilkan C2-itemset, C2 terdiri dari $\binom{|L1|}{2}$ 2-itemsets. Pencarian kombinasi C2 tersebut bisa di representasikan dalam bentuk akar pohon seperti gambar 3 berikut.

Hasil dari pembangkitan C2-itemset dari L1-itemset adalah :

$C2 = L_1 \text{ join } L_1 = \{2\},\{3\},\{6\},\{8\},\{9\} \bowtie \{2\},\{3\},\{6\},\{8\},\{9\} = \{\{2,3\}, \{2,6\}, \{2,8\}, \{2,9\}, \{3,6\}, \{3,8\}, \{3,9\}, \{6,8\}, \{6,9\}, \{8,9\}\}$ sebanyak 10 itemsets.

Dari tiap-tiap kombinasi 2-itemset maka C2-itemsets maka nilai support dari C2-itemset dilihat pada tabel 7 berikut :

Tabel 8. C2-Itemsets Transaksi D dengan nilai support

No	C2-Itemsets	Support
1	{2,3}	2
2	{2,6}	0
3	{2,8}	1
4	{2,9}	1
5	{3,6}	0
6	{3,8}	0
7	{3,9}	1
8	{6,8}	2
9	{6,9}	2
10	{8,9}	2

Eliminasi setiap frequent itemset C2 dengan membandingkan nilai $min_sup = 2$ dengan jumlah support masing-masing frequent itemset, Jika lebih kecil dari min_sup maka eliminasi dari keanggotaan C2. Itemset {2,6}, {2,8}, {2,9}, {3,6}, {3,8}, {3,9} di-eliminasi karena hanya memiliki nilai support TIDAK ≥ 2 dan itemset {2,3},{6,8},{6,9},{8,9} menjadi L2-itemset sebagai 2-itemset yang paling sering muncul atau frequent. Hasil dari L2-itemsets dapat dilihat pada tabel 8.

Tabel 9. L2-Itemsets Apriori Data Transaksi D

No	Itemsets	Support
1	{2,3}	2
2	{6,8}	2
3	{6,9}	2
4	{8,9}	2

Selanjutnya adalah mencari $K+1 = 3$, C3-Itemsets. Proses nya sama seperti K1,K2 sehingga didapatkan hasil seperti berikut :
 $C3 = L_2 \text{ join } L_2 = \{\{2,3\}, \{6,8\}, \{6,9\}, \{8,9\}\}$
 $\bowtie \{\{2,3\}, \{6,8\}, \{6,9\}, \{8,9\}\}$
 $= \{\{2,3,6\}, \{2,3,8\}, \{2,3,9\}, \{2,6,8\}, \{2,6,9\}, \{2,8,9\}, \{3,6,8\}, \{3,6,9\}, \{3,8,9\}, \{6,8,9\}\} = 10 \text{ itemsets}$

Hasil pembangkitan kandidat 3-itemset atau C3 berdasarkan L2-itemset dari tabel 9 sebelumnya dapat dilihat pada tabel 3.14 :

Tabel 10. C3-Itemsets Apriori

No	Itemsets	Support
1	{2, 3, 6}	0
2	{2, 3, 8}	0
3	{2, 3, 9}	0
4	{2, 6, 8}	0
5	{2, 6, 9}	0
6	{2, 8, 9}	0
7	{3, 6, 8}	0
8	{3, 6, 9}	0
9	{3, 8, 9}	0
10	{6, 8, 9}	1

Dari dataset transaksi D, maka hasil akhirnya didapatlah frequent itemset terbesar pada posisi K3, yaitu {6, 8, 9} sejumlah 1 itemsets.

Selanjutnya adalah proses pencarian k-itemset berhenti karena tidak ada kombinasi sepanjang K4 di temukan pada datasets. Maka hasil akhirnya adalah
 L1-itemset = {2},{3},{6},{8},{9} = 5
 L2-itemset = {2,3}, {6,8}, {6,9}, {8,9} = 4
 L3-itemset = {6, 8, 9} sebagai itemset terbesar.

HASIL DAN PEMBAHASAN

Dari hasil ujicoba menggunakan program maka dataset *kosarak.dat* dan *mushrooms.dat* dapat dilihat pada tabel berikut :

Tabel 11. L1-Itemsets data *kosarak.dat* minsup ≥ 0.010

No	Frequent Items & Support	No	Frequent Items & Support
1	1 #SUP: 197522	28	148 #SUP:
2	2 #SUP: 42927	29	69922
3	3 #SUP: 450031	30	155 #SUP:
4	4 #SUP: 78097	31	11494
5	6 #SUP:	32	205 #SUP:
6	601374	33	21373
7	7 #SUP: 86898	34	215 #SUP:
8	11 #SUP:	35	21592
9	364065	36	218 #SUP:
10	14 #SUP: 11607	37	88598
11	25 #SUP: 12668	38	229 #SUP:
12	27 #SUP: 72134	39	10034
13	28 #SUP: 10111	40	254 #SUP:
14	32 #SUP: 11069	41	10077
15	40 #SUP: 24082	42	269 #SUP:
16	49 #SUP: 12702	43	18343
17	55 #SUP: 65412	44	273 #SUP:
18	56 #SUP: 15752	45	17177
19	64 #SUP: 48262	46	278 #SUP:
20	69 #SUP: 22631	47	20114
21	73 #SUP: 18749	48	294 #SUP:
22	77 #SUP: 43454	49	32247
23	83 #SUP: 32212	50	303 #SUP:
24	87 #SUP: 18484	51	21246
25	90 #SUP: 22062	52	314 #SUP:
26	91 #SUP: 12248	53	14842
27	135 #SUP:	54	316 #SUP:
	18313		27852
	136 #SUP:		361 #SUP:
	23679		10122
	138 #SUP:		364 #SUP:
	35127		12167
			378 #SUP:
			11508
			423 #SUP:
			10967
			438 #SUP:
			18910
			446 #SUP:
			23270
			490 #SUP:
			23225
			504 #SUP:
			14546
			512 #SUP:
			12921
			667 #SUP:
			10616
			737 #SUP:
			17645
			897 #SUP:
			14297
			987 #SUP:
			17804

Tabel 12. *L2-Itemsets* data *kosarak.dat* minsup ≥ 0.010

No	Frequent Items & Support
1	1 6 #SUP: 25466
2	4 6 #SUP: 13601
3	11 27 #SUP: 24917
4	11 218 #SUP: 23678

Hal yang sama dilakukan pada minimum support 0.020 dan 0.030 sehingga didapat hasil *frequent itemsets* L1 dan L2-*itemsets* pada min sup 0.010 adalah :

Kandidat = 1.484

L1 = 54 items = 6 #SUP: 601.374 terbesar

L2 = 4 itemsets = 6, 11 #SUP: 324.013 terbesar

Pada hasil selanjutnya dapat dilihat pada tabel berikut :

Tabel 13. K-itemset *kosarak.dat* dengan apriori minsup 0.010 - 0.030

No	Min Supp	K	L1	L2	L-Itemsets
1	0.030	136	16	-	16
2	0.020	378	27	3	30
3	0.010	148	54	4	58

Sedangkan pada dataset *mushrooms.dat* hasil uji data dengan program dengan minimum support 0.010-0.030, lebih lengkapnya dapat dilihat pada tabel berikut :

Tabel 14. K-itemset *mushrooms.dat* dengan apriori minsup 0.010 - 0.030

No	Min Supp	K-Itemsets	L-Itemsets
1	0.030	3.059	2.735
2	0.020	54.522	53.583
3	0.010	577.457	574.431

Hasil dari penggunaan sumber daya dapat dilihat pada tabel dibawah ini.

Tabel 15. Hasil penggunaan sumber daya pada *apriori* minsup 0.010-0.030

No	Min Sup	Kosarak		Mushroom	
		Time (ms)	Memory (mb)	Time (ms)	Memory (mb)
1	0.030	5.984	348	4.719	92
2	0.020	13.656	490	60.989	158

3	0.010	41.532	685	529.323	269
---	-------	--------	-----	---------	-----

Dari hasil ujicoba datatersebut dapat disajikan dalam bentuk grafik.

Gambar 3. Grafik Kandidat dan *Frequent itemsets* dataset *kosarak.dat*

Gambar 4. K-Itemsets datasets *mushrooms.dat*

Gambar 6 L-Itemsets datasets *mushrooms.dat*

Grafik hasil perbandingan penggunaan sumber daya bisa dilihat di bawah ini :

Gambar 5. Penggunaan sumberdaya dalam satuan waktu miliseconds (ms)

Gambar 6. Penggunaan memori dalam satuan megabyte (mb)

SIMPULAN

Beberapa kesimpulan yang dapat diambil dari analisis diatas adalah Jika jumlah minimum support kecil maka proses pencarian apriori semakin lambat dan jumlah *K-itemsets* semakin banyak. Datasets *kosarak.dat* memiliki *frequent itemsets* yang lebih sedikit jika dibandingkan dengan *mushrooms.dat* waktu yang dibutuhkan apriori untuk mencari *frequent itemsets* pada *kosarak.dat* lebih sedikit dibandingkan dengan datasets *mushrooms.dat* yang membutuhkan waktu yang lebih lama.

DAFTAR PUSTAKA

- Agrawal, R., Imielinski, T. & Swami, A.1993. Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD Conference*, pp. 207–216.
- Agrawal, R. & Srikant, R.1994. Fast algorithm for mining association rules. *Proceedings of the 20th VLDB Conference*, pp. 487–499.
- Bhandari, A., Gupta, A., & Dasa, D.2015. Improved apriori algorithm using frequent pattern tree for real time applications in data mining. *International Conference on Information and Communication Technologies (ICICT)*.46:644-651.
- Camp, R."The search for industry best practices that lead to superior performance. *Productivity Press*".1989
- Darshan. 2014. Improved Apriori Algorithm for mining Association Rules.*International*

- Journal of Information Technology and Computer Science (IJITCS'07)*, pp.15-23.
- Frequent Itemset Mining Dataset Repository by IBM Almaden Quest Research Group (Online)<http://fimi.ua.ac.be/data/>.(07 April 2016).
- Gorunescu, F.2011. Data Mining :Concepts, models and technique.Springer: Verlag Berlin Heidelberg.
- Han, J., Kamber, M. & Pei, J. 2012. Data Mining: Concept and techniques.3rd Edition. Simon Fraser University.Morgan Kauffman Publisher:Amsterdam.
- Mangla, V., Sarda,C. & Madra,S.2013. Improving the efficiency of Apriori Algorithm in Data Mining. *International Journal of Engineering and Innovative Technology (IJEIT)* 3(3):393-396
- Singh, J., Ram, H. & Sodhi,J.S.2013.Improving Efficiency of Apriori Algorithm Using Transaction Reduction.*International Journal of Scientific and Research Publications (IJSRP)* 3(1):1-4.