Semantika
Seminar Nasional Teknologi Informatika

Analisis Efisiensi Algoritma Data Mining

Rizki Muliono
Teknik Informatika, Universitas Medan Area
rizkimuliono@gmail.com

Abstrak

Datamining adalah kaidah ilmu yang semakin berkembang pesat saat ini dan semakin bahkan harus digunakan dalam teknologi saat ini dan seterusmya. Terdapat beberapa teknik dalam datamining yang bisa digunakan untuk mengumpulkan informasi yang ada pada database atau dataset. proses tersebut diantaranya dapat berupa pencarian patern, pola asosiasi, kombinasi item yang kesemuanya adalah mencari frequent itemset yaitu kombinasi item yang sering muncul pada database atau dataset. kombinasi tersebut adalah item A dengan Item B yang saling berkorelasi seterusnya dengan item C sampai pada jumlah K-itemset atau deret kombinasi terbesar yang di temukan. Frequent itemset adalah pola yang sangat populer dalam teknik datamining, hasil dari teknik ini dapat dijadikan sebagai hasil yang akurat pada kaidah analisis pasar atau di kenal dengan Market Basket Analysis dalam pengambilan keputusan. Apriori adalah teknik dasar dari datamining yang dapat menemukan kombinasi Frequent itemset namun algoritma tersebut masih memiliki kelemahan-keleman dalam proses hasil pencarian dan performa terhadap berbagai macam model dataset dan sumberdaya yang di habiskan sehingga pakar telah melakukan pendekatan dengan membuat improvment pada apriori tersebut namun masing-masing teknik baru tersebut perlu di analisa dan di bandingkan dari segi performa dan kaidah lainnya yang lebih spesifik. Pada tulisan ini akan di bahas analisis performa dari algoritma yang telah di improvment oleh pakar diantaranya Apriori-Close dan Apriori FP-Growth dan di ujicobakan dengan menggunakan dataset dari FIMI repository dataset yaitu kosarak dataset. Hasil analisa yang akan di tampilkan adalah hasil frequent itemset yang di temukan masing-masing algoritma tersebut dengan ujicoba threshold atau nilai ambang batas minimum support 0.10 - 0.50, kemudian waktu dan memori yang di gunakan di sajikan dalam bentuk tabel dan grafik. Diharapkan dengan ujicoba ini dapat menemukan informasi perbedaan yang lebih spesifik dari algoritma-algoritma tersebut sehingga menjadi hasil analisis yang dapat di pakai untuk studi-studi litertur

Keywords : Data mining, Frequent Itemsets, Apriori-Close, FP-Growth.

I. PENDAHULUAN

Frequent itemset adalah masalah dan kepentingan yang mendasar pada konsep aplikasi didalam data mining, termasuk mencari dan menemukan aturan asosiasi, frequent itemset terkuat, hubungan kolerasi antar itemset, multi-dimensional pattern dan lainnya[1]. Variasi algoritma mencari pola atau pattern-mining yang paling banyak di gunakan adalah berdasarkan dari algoritma Apriori [2].

Apriori menggunakan sebuah bottom-up, Pencarian pertama yang menghitung setiap

Semantika
Seminar Nasional Teknologi Informatika

frequent itemset. Apriori menggunakan komparasi ke bawah dari setiap properti itemset untuk memangkas ruang pencarian - properti yang semua subset dari *frequent itemset* tunggal yang sering muncul. Jadi hanya k-itemset yang sering mucul digunakan untuk membuat kandidat (k+1)-*Itemset* sampai pencarian di akhiri jika tidak ada lagi jumlah kombinasi K+1 di temukan di dalam database atau dataset [3].

A. Latar Belakang

Algoritma Apriori secara mendasar menunjukkan kinerja yang baik dengan pola dataset seperti dataset market basket, dimana pola *frequent itemsets* sangat pendek. Namun, dengan dataset padat seperti telekomunikasi Dan data sensus, dimana ada banyak, pola kombinasi dan komparasi yang sering terjadi, kinerja algoritma ini menurun dengan luar biasa [4, 5]. Penurunan ini disebabkan oleh beberapa alasan berikut: algoritma ini melakukan banyak hal didalam komparasi berulang pada database untuk menemukan frequent itemset terpanjang [6]. Kelemahan pada algoritma apriori tersebut telah di selesaikan bebarapa pakar yang menghasilkan pengembangan kemampuan dari kinerja algoritma tersebut sehingga lebih efektif yaitu algoritma *AprioriClose* dan *FP-Growth*. Namun secara liratur kedua algoritma ini perlu diuji kembali kinerja dan efisiensi nya dengan beberapa pola dataset untuk mengetahui tingkat efisiensinya, kedua algoritma tersebut akan di uji pada dataset kosarak dan mushroom dengan nilaii minimum support adalah 0.10 – 0.50 hasilnya akan di tampilkan dalam bentuk tabel dan diagram performa.

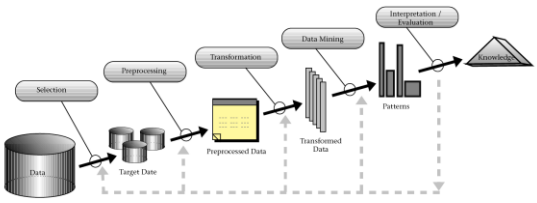
II. TINJAUAN PUSTAKA

A. Data mining

Data mining dalam defenisinya sebagai satu set teknik yang secara otomatis digunakan untuk mengeksplorasi data secara menyeluruh dan menghasilkan relasi-relasi yang kompleks pada set data. Set data yang dimaksud di sini adalah set data yang berbentuk tabulasi, seperti yang banyak diimplementasikan dalam teknologi manajemen basis data relasional. Data mining menggunakan pendekatan discovery-based dimana pencocokan pola (pattern-matching) dan algoritma-algoritma yang lain digunakan untuk menentukan relasi-relasi kunci di dalam data yang diekplorasi [7].

Dengan tersedianya basis data dalam kualitas dan ukuran yang memadai, teknologi data mining memiliki kemampuan-kemampuan sebagai berikut:

1. Mampu mencari prediksi tren dan sifat-sifat bisnis. Data mining mengotomatisasi proses pencarian informasi pemprediksi di dalam basis data yang besar. Contoh dari masalah prediksi ini misalnya target pemasaran, peramalan kebangkrutan dan bentuk bentuk kerugian lainnya.
2. Mengotomatisasi penemuan pola-pola yang tidak diketahui sebelumnya. Contoh dari penemuan pola ini adalah analisis pada data penjulan ritel untuk mengidentifikasi produk-produk, yang kelihatannya tidak berkaitan, yang seringkali dibeli secara bersamaan oleh kustomer. Contoh lain adalah pendeteksian transaksi palsu dengan kartu kredit dan identifikasi adanya data anomali yang dapat diartikan sebagai data salah ketik (karena kesalahan operator) [8].



Gambar 1. Proses Knowledge Data Discovery (KDD) [4]

Dari gambar diatas kita dapat mengetahui proses dasar dari datamining diataranya : (1)Mengerjakan sejumlah besar data. (2)Diperlukan efesiensi berkaitan dengan volume data. (3)Mengutamakan ketetapan/keakuratan. (4)Membutuhkan pemakaian bahasa tingkat tinggi. (5)Menggunakan beberapa bentuk dari pembelajaran otomatis. (6)Menghasilkan hasil yang menarik.

B. *Frequent Itemsets Mining*

Setiap himpunan bagian *itemsets* yang suportnya lebih tinggi dari nilai minimum yang ditetapkan disebut *nimimum 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 s*upport* 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 →CD jika rasio dari *support*(ABCD) terhadap *support*(AB) sedikitnya sama dengan *minimum confidence*. Aturan ini memiliki minimum support karena ABCD adalah *frequent*.

$$\text{Support (A)} = \left(\frac{\text{jumlah transaksi mengandung A}}{\text{Total transaksi}}\right) x100\%$$

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

C. *Association rule*

Association rule adalah salah satu teknik utama atau prosedur dalam *Market Basket Analysis* untuk mencari hubungan antat item dalam suatu *data set* dan menampilkan dalam bentuk *association rule* [10]. 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].

Misalkan I adalah itemset, dan D adalah database transsaksi, dimana setiap transaksi memiliki transaksi id (tid) dan terdiri dari itemset. Semua set transaksi tid kita dengan notasi T. Sebuah set $X \subseteq I$ juga di sebut sebagai itemset, dan sebuah set $Y \subseteq T$ juga disebut tid set. Sebuah itemset dengan k items di sebut k-itemset. Kombinasi sebuah itemset {A, C, W} sebagai ACW, dam tid set {2, 4, 5} sebagai 245. Dari sebuah itemset X, kita notasikan tidset sebagai (X), dan semua set tids yang mengandung X adalah sebuah subset. Untuk sebuah tidset Y, kita notasikan memiliki hubungan itemset sebagai i(Y), dan set dari items bersangkutpaut terhadap ke semua tid Y. Kita dapat simpulkan $t(X) = x \in X\ t(x)$, dan $i(Y) = y \in Y\ i(y)$. Sebagai contoh, t(ACW) = t(A) ∩ t(C) ∩ t(W) = 1345 ∩ 123456 ∩ 12345 = 1345 dan i(12) = i(1) ∩ i(2) = ACTW ∩ CDW = CW. Kita gunakan notasi X × t(X) untuk mengkaitkan itemset-tidset yang sesuai, kita notasikan sebagai IT-pair. Nilai support [2] sebuah itemset X, dinotasikan σ(X), yaitu jumlah kombinasi itemset dari transaksi yang memiliki sebuah subset, yaitu σ(X) = |t(X)|. Sebuah *itemset* dikatakan *frequent* jika nilai support lebih dari atau sama dengan nilai yang di tentukan sebagai minimum nilai support (*min sup*), misal jika σ(X) ≥ *min sup*. Sebuah *frequent itemset* dikatakan maksimal jika tidak ada lagi sebuah subset dari frequent itemset yang lainnya. sebuah frequent itemset X berakhir jika tidak ditemukan lagi di dalam database.

D. *Data Benchmark*

Model *Benchmarking* adalah pendekatan yang secara terus menerus mengukur dan membandingkan produk barang dan jasa, dan proses-proses terhadap standarisasi yang ditetapkan yang dianggap unggul dalam bidang tersebut. Dengan melakukan atau melalui *benchmarking*, suatu organisasi dapat mengetahui berapa jauh mereka dapat membandingkan data dengan yang terbaiknya[11]. Pada penelitian ini menggunakan data *benchmark* dari bersumber dari *FIM Dataset Repository* yang di publikasikan oleh *IBM Almaden Quest Research Group* bersumber dari *http://fimi.ua.ac.be/data/*.[12]

E. *Dataset Kosarak*

kosarak.dat adalah sekumpulan *clik-stream* dari portal berita di Hungaria, dengan tipe data *sparce* yaitu berupa angka-angka dengan jumlah baris sebanyak 990.002 dan 41.270 items berbeda didalamnya. Dataset ini sudah banyak di gunakan oleh peneliti sebagai percobaan dalam mengembangkan ilmu data mining.

TABEL 1. *SAMPEL ISI DATA KOSARAK.DATA*

No	Isi Data Transaksi
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

Semantika
Seminar Nasional Teknologi Informatika

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 35 36 37
13.	6 2 38
14.	39 11 27 1 40 6 41 42 43 44 45 46 47 3 48 7 49
15.	50 51
16.	52 6 3 53
17.	54 1 6 55
18.	11 6 56 57 58 59 60 61 62 63 64
…	…

F. Algoritma Apriori

Dalam algoritma Apriori, item diurutkan berdasarkan urutan leksikografis. *Frequent itemset* dicari dan dihitung secara iteratif, dalam urutan menaik sesuai jumlah ukuran datanya. Prosesnya membutuhkan sebanyak k-iterasi, dimana k adalah ukuran *Itemset frequent* terbesar untuk setiap iterasi i k, database dibaca sekali dan menghituang jumlah frequent Itemset yang ada. Pada Iterasi pertama menghitung set L1 dari frequent 1-itemsets. Selanjutnya iterasi terdiri dari dua fase. Pertama, satu set Ci dari kandidat i-itemset dibuat oleh frequent yang digabung (i1) -sepertinya di Li1 Ditemukan pada iterasi sebelumnya. Fase ini diwujudkan oleh fungsi *Apriori-Gen* yang membangkitkan kombinasi. Selanjutnya, database dipindai untuk menentukan support kandidat di Ci dan frequent i-itemset diekstraksi dari kandidat. Proses ini diulang sampai tidak ada lagi kandidat yang bisa dihasilkan [14]. Berikut ini adalah pseudo code dari algoritma apriori.

1) L1 = {Large 1-itemsets};
2) for (k=2; Lk1 6= ;; k++) do begin
3) Ck = Apriori-Gen(Lk1); // Generates candidates k-itemsets
4) forall transactions t 2 D do begin
5) Ct = Subset(Ck ,t); // Candidates contained in t 6) forall candidates c 2 Ct do
7) c.count++;
8) end
9) Lk = { c 2 Ck | c.count minsupport }
10) end
11) Answer = S k Lk;

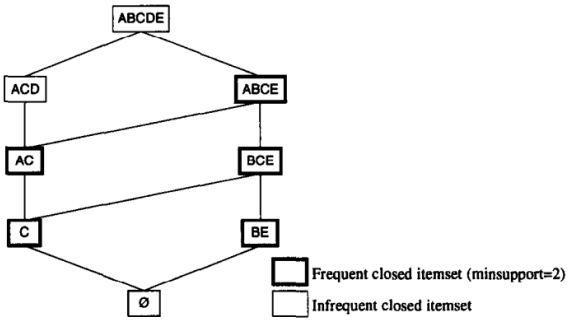
Algoritma 1 : Pseudocode algoritma Apriori [2]

G. Algoritma *Apriori-Close*

Itemset terdekat pada sebuah dataset dapat diambil dari *frequent itemset* pad saat terjadinya proses komparasi, namun menghasilkan solusi yang lebih cepat, jika menggunakan pencarian dengan algoritma FIM. *Apriori-Close* melakukan pemangkasan kandidat yang sering muncul yang dilakukan dengan satu langkah, di mana himpunan bagian dari kandidat yang frequent diperiksa. Secara default semua subset ditandai sebagai memiliki kedekatan, yang diubah jika nilai support subset sama dengan kandidat yang benar-benar diperiksa saat komparasi. Akibatnya, di *Apriori-Close* semua himpunan bagian dari kandidat frequent yang dihasilkan, dimana banyak proses di skema trie. Proses komparasi yang sia-sia akan dihindari jika penyaringan itemset terdekat dilakukan di fase saat pembangkitan kandidat dan pemangkasan dengan menggunakan metode pendekatan terapan. Dengan metode ini, subset sudah ditentukan, sehingga memeriksa kesetaraan nilai support tidak memerlukan komparasi ekstra [13].

Algoritma *Apriosi-Close* secara fundamental berbeda dari algoritma yang ada, karena didasarkan pada Pemangkasan pada itemset terdekat untuk menemukan frequent itemset. Itemset close adalah sebuah set item maksimal yang sering mncul. Misalnya, di database D, itemset {B, C, E} adalah itemset terdekat karena ini adalah rangkaian item yang paling umum yang diketahui untuk objek {2,3,5}.

{B, C, E} disebut frequent closed itemset untuk minsupport = 2 sebagai support ({ B, C, E}) = ||{2,3,5}|| = 3 ≥ ***minsupport***. Dalam database, ini berarti 60% pelanggan (3 pelanggan dari total 5) membeli paling banyak item B, C, E. Itemset {B, C} bukan itemset terdekat, semenjak bukan kelompok barang maksimal yang umum untuk beberapa objek: semua pelanggan membeli barang B dan C juga membeli item E.



Gambar 2. Closed Itemset Lattice dari database D [14]

(1) Prosesnya adalah menemukan semua itemset terdekat yang sering mucul di D, yaitu itemset yang terdekat dan memiliki nilai *min_support* lebih besar atau sama dengan *min_support*.
(2) Turunkan semua *frequent itemset* dari *frequent itemsset* yang sering ditemukan pada fase 1. Fase ini akan menghasilkan semua subset dari itemset terdekat yang maksimal dan menurunkannya berdasarkan dari *min_sup* itemset terdekat yang *frequent*.
(3) Setiap *frequent itemset* yang ditemukan di fase 2, buat semua aturan asosiasi yang bisa diturunkan yang memiliki nilai *confidece* lebih besar atau sama dengan *minconfidence*.

Apriori-Close memiliki pseudocode seperti dibawah ini :

1) **generators in** $FCC_1 \leftarrow \{1\text{-itemsets}\}$;
2) **for** $(i \leftarrow 1;\ FCC_i.\text{generator} \neq \emptyset;\ i{+}{+})$ **do begin**
3) **closures in** $FCC_i \leftarrow \emptyset$;
4) **supports in** $FCC_i \leftarrow 0$;
5) $FCC_i \leftarrow$ **Gen-Closure**$(FCC_i)^{\dagger}$; // Produces generator closures
6) **forall candidate closed itemsets** $c \in FCC_i$ **do begin**
7) **if** $(c.\text{support} \geq minsupport)$ **then** // If c is frequent
8) $FC_i \leftarrow FC_i \cup \{c\}$; // Append c to FC_i
9) **end**
10) $FCC_{i+1} \leftarrow$ **Gen-Generator**(FC_i); // Creates generators of iteration $i{+}1$
11) **end**
12) **Answer** $FC \leftarrow \bigcup_{j=1}^{j=i-1} \{FC_j.\text{closure}, FC_j.\text{support}\}$;

Algoritma 2 : Pseudocode algoritma *Apriori-Close* [14]

H. *Algoritma FP-Growth*

FP-growth adalah salah satu algoritma yang tercepat untuk solusi pencarian frequent item set didalam dataminig [15]. Algoritma ini mencari itemset tanpa melakukan Generasi kandidat itemset dan menggunakan struktur data pohon atau (FP-Tree).
Metode FP-Growth dapat dibagi menjadi 3 tahapan utama yaitu sebagai:
1. Tahap pembangkitan conditional *pattern base*,
2. Tahap pembangkitan conditional *FP-Tree*, dan
3. Tahap pencarian frequent itemset.

Ketiga tahap tersebut merupakan langkah yang akan dilakukan untuk mendapat *frequent itemset*.

Input : FP-Tree Tree
Output : Rt Sekumpulan lengkap pola frequent
Methode : FP-Growth (Tree, null)
Procedure : FP-Growth (Tree, α)
{
01: if Tree mengandung single path P;
02: then untuk tiap kombinasi (dinotasikan β) dari node-node dalam path do
03: bangkitkan pola β α dengan support dari node-node dalam path do β;
04: else untuk tiap a1 dalam header dari tree do
{
05: bangkitkan pola
06: bangun β = a1 α dengan support = a1l support 07: if Tree β =∅
08: Then panggil FP-Groeth(Tree, β)
}
}

Adapun *FP-tree* adalah sebuah pohon dengan definisi sebagai berikut:
- *FP-Tree* dibentuk oleh sebuah akar yang diberi label *null*, sekumpulan akar pohon yang beranggotakan item-item tertentu, dan sebuah tabel *frequent header*.
- Setiap simpul dalam *FP-tree* mengandung tiga informasi penting, yaitu label item, menginformasikan jenis item yang direpresentasikan simpul tersebut, *support count*, merepresentasikan jumlah lintasan transaksi yang melalui simpul tesebut, dan pointer penghubung yang menghubungkan simpul-simpul dengan label item sama antar lintasan, dintandai dengan garis panah putus-putus.

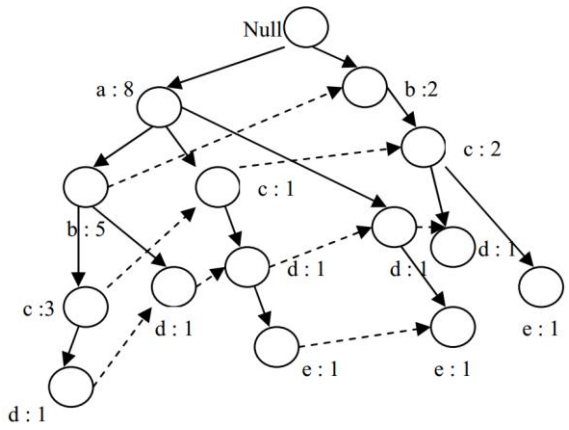
Misalkan data transaksi seperti pada tabel berikut :

TABEL 2. DATA TRANSAKSI

TID	Transaksi
1	a,b
2	b,c,d,g,h
3	a,c,d,e,f
4	a,d,e
5	a,b,z,c
6	a,b,c,d
7	a,r
8	a,b,c
9	a,b,d
10	b,c,e

Data transaksi tersebut jika menggunakan algoritma *FP-growth* dengan minsup ≥ 2 maka dapat dibentuk selanjutnya menjadi sebuah *FP-tree*.

Semantika
Seminar Nasional Teknologi Informatika



Gambar 3. Hasil Pembentukan FP-Tree setelah pembacaan TID 10

Setelah memeriksa *frequent itemset* untuk beberapa akhiran (*suffix*), maka didapat hasil yang dirangkum dalam tabel berikut:

TABEL 3. FREQUENT ITEMSETS

Suffix	*Frequent Itemset*
e	{e},{d,e},{a,d,e},{c,e},{a,e}
d	{d},{c,d},{b,c,d},{a,c,d},{b,d}, {a,b,d},{a,d}
c	{c},{b,c},{a,b,c},{a,c}
b	{b},{a,b}
a	{a}

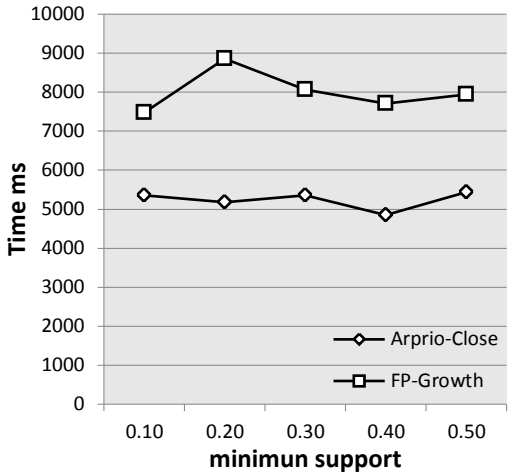
III. HASIL DAN ANALISA

Hasil ujicoba dataset *kosarak.dat* dengan algotima *Apriori-Close* dan *FP-Growth* adalah :

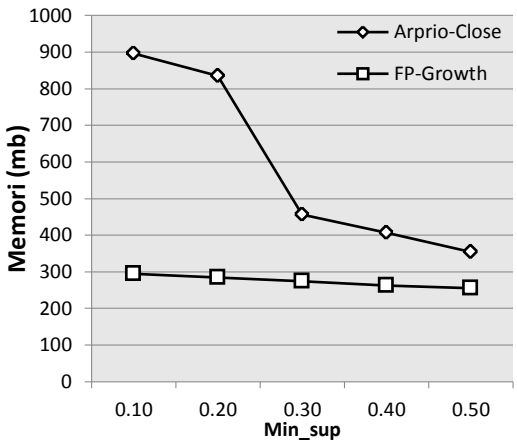
TABEL 4. ANALISA HASIL PERFORMA

Minsup	*Apriori-Close*		*FP-Growth*	
	ms	*mb*	*ms*	*mb*
0.50	5443	355	7937	256
0.40	4846	407	7701	262
0.30	5357	456	8064	284
0.20	5172	835	8859	295
0.10	5360	897	7469	274

Dari hasil tabel diatas dapat kita sajikan data kedalam bentuk grafik performa waktu proses (ms) dan memori yang digunakan (mb).



Gambar 4. Grafik Performa Waktu proses Algoritma



Gambar 5. Grafik Performa penggunaan memori

IV. KESIMPULAN

Dari hasil ujicoba kedua algoritma pada penelitian ini maka didapatlah beberapa kesimpulan :

4. Algotima *Apriori-Close* lebih cepat dalam proses pencarian frequent itemset dibandingkan dengan *FP-Growth* pada kasus data *kosarak.dat* atau *Apriori-Close* unggul dalam waktu proses.
5. Memori yang digunakan *Apriori-Close* lebih besar jika dibandingkan dengan yang digunakan algritma *FP-Growth* atau *Apriori-Close* lebih boros sumberdaya memori dibanding *FP-Growth.*

6. *FP-Growth* pada penggunaan memori semakin kecil nilai minsup memori yang dipakai semakin besar, sedang *Apriori–Close* sedikit signifikan.
7. Masalah waktu *Apriori-Close* lebih efisien.
8. Masalah memori *FP-Growth* lebih efisien.

DAFTAR PUSTAKA

[1] J. Han and M. Kamber. Data Mining: Concepts and Techniuqes. Morgan Kaufmann Publishers, San Francisco, CA, 2001.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In U. Fayyad and et al, editors, Advances in Knowledge Discovery and Data Mining, pages 307–328. AAAIPress, Menlo Park, CA, 1996.

[3] Zaki, M. J., & Hsiao, C. J. CHARM: An efficient algorithm for closed itemset mining. In Proceedings of the 2002 SIAM international conference on data mining (pp. 457-473). Society for Industrial and Applied Mathematics. (April,2002)

[4] S. Brin, R. Motwani, J. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In ACM SIGMOD Conf. Management of Data, May 1997

[5] J. S. Park, M. Chen, and P. S. Yu. An effective hash based algorithm for mining association rules. In ACM SIGMOD Intl. Conf. Management of Data, May 1995.

[6] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In 21st VLDB Conf., 1995

[7] V.S. Moertini. Datamining sebagai solusi bisnis. Integral. vol. 7 no. 1, April 2002

[8] Fayyad U.M., Piatetsky-Shapiro G., Smyth P., Uthurusamy R., Advance in Knowledge Discovery and Data Mining, MIT Press, Cambridge MA, 1996

[9] Han, J., Kamber, M. & Pei, J. Data Mining: Concept and techniques.3rd Edition. Simon Fraser University.Morgan Kauffman Publisher:Amsterdam, 2012.

[10] Gorunescu, F.Data Mining : Concepts, models and technique.Springer: Verlag Berlin Heidelberg.2011.

[11] Camp, R.The search for industry best practices that lead to superior performance. Productivity Press.1989

[12] Frequent Itemset Mining Dataset Repository by IBM Almaden Quest Research Group (Online) http://fimi.ua.ac.be/data/. (07 April 2016).

[13] Bodon, Ferenc, and Lars Schmidt-Thieme. The relation of closed itemset mining, complete pruning strategies and item ordering in apriori-based fim algorithms. European Conference on Principles of Data Mining and Knowledge Discovery. Springer Berlin Heidelberg, 2005.

[14] Pasquier, N., Bastide, Y., Taouil, R., & Lakhal, L. Pruning closed itemset lattices for association rules. In BDA'1998 international conference on Advanced Databases (pp.177-196). Oktober 1998.

[15] Borgelt, Christian. An Implementation of the FP-growth Algorithm. Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations. ACM, 2005.