Class Association Rule Pada Metode *Associative Classification*

E. Karyawati dan E. Winarko

Abstract— *Frequent patterns (itemsets) discovery is an important problem in associative classification rule mining. Differents approaches have been proposed such as the Apriori-like, Frequent Pattern (FP)-growth, and Transaction Data Location (Tid)-list Intersection algorithm. This paper focuses on surveying and comparing the state of the art associative classification techniques with regards to the rule generation phase of associative classification algorithms. This phase includes frequent itemsets discovery and rules mining/extracting methods to generate the set of class association rules (CARs). There are some techniques proposed to improve the rule generation method. A technique by utilizing the concepts of discriminative power of itemsets can reduce the size of frequent itemset. It can prune the useless frequent itemsets. The closed frequent itemset concept can be utilized to compress the rules to be compact rules. This technique may reduce the size of generated rules. Other technique is in determining the support threshold value of the itemset. Specifying not single but multiple support threshold values with regard to the class label frequencies can give more appropriate support threshold value. This technique may generate more accurate rules. Alternative technique to generate rule is utilizing the vertical layout to represent dataset. This method is very effective because it only needs one scan over dataset, compare with other techniques that need multiple scan over dataset. However, one problem with these approaches is that the initial set of tid-lists may be too large to fit into main memory. It requires more sophisticated techniques to compress the tid-lists.*

Kata Kunci— *frequent itemset, CAR, apriori, FP-growth, tid-list, associative classification*

Eka Karyawati, *Jurusan Ilmu Komputer, Fakultas matematika dan Ilmu Pengetahuan Alam, Universitas Gadjah Mada*. e-mail: ekakaryawati@yahoo.com

Edi Winarko, Jurusan Ilmu Komputer, Fakultas matematika dan Ilmu Pengetahuan Alam, Universitas Gadjah Mada. e-mail: ewinarko@yahoo.com

1. PENDAHULUAN

Associative classification (AC) adalah cabang dalam *data mining* yang memanfaatkan metode *association rule discovery* dalam masalah klasifikasi. Beberapa studi memberikan bukti bahwa teknik AC dapat memberikan model klasifikasi yang lebih akurat dibandingkan dengan teknik-teknik klasifikasi tradisional seperti *decision tree*, *rule induction*, dan pendekatan probalistik [12].

Sama seperti *association rule mining*, masalah mendasar pada AC adalah metode dalam membangkitkan *frequent itemset* dari data yang berukuran besar. Tugas ini memakan waktu yang sangat besar karena data transaksi harus discan berulang kali untuk mencari *frequent itemset*. Banyak riset mengusulkan teknik-teknik untuk memperbaiki metode dalam membangkitkan *frequent itemset*. Membangkitkan *frequent itemset* merupakan satu bagian dari fase pertama dalam metode AC, yang disebut fase *rule generation*. Fase *rule generation* meliputi metode untuk menemukan *frequent itemset* dan metode untuk mengekstrak rule dengan tujuan membangkitkan himpuan *class association rule* (CAR). Fase kedua pada metode AC adalah fase *building classifier* dan fase klasifikasi. Fase *building classifier* mencoba menghilangkan *redundant rule* (*pruning*) dan mengurutkan rule yang berguna (*useful*) dalam urutan yang tepat (*ranking*). Fase klasifikasi adalah fase untuk memprediksi atau mengklasifikasikan data yang belum dikategorikan dalam kelas tertentu atau belum berlabel.

Paper ini difokuskan pada survey dan pembandingan metode associative classification dengan fokus pada fase pertama dari algoritma *associative classification*.

Ada banyak metode *associative classification* mining seperti CBA, CMAR, MCAR, L3G, dan sebagainya. Metode-metode tersebut secara umum menggunakan beberapa pendekatan berbeda dalam pembangkitan rule (CAR). Penulis membandingkan tiga jenis pendekatan yang berbeda: metode *Apriori*, metode *Frequent*

Pattern (FP)-Growth Tree, dan metode *Transaction Data Location (Tid)-list Intersection*.

2. METODE PENELITIAN

Metode penelitian yag digunakan adalah studi literatur dengan mereview beberapa jurnal yang terkait dengan metode *associative classification*, khususnya pada fase *rule generation*.

Penulis membandingkan tiga jenis pendekatan yang berbeda: metode *Apriori*, metode *Frequent Pattern (FP)-Growth Tree*, dan metode *Transaction Data Location (Tid)-list Intersection*.

2.1 Associative Classification

Associative classification adalah kasus khusus dari *association rule mining* dimana hanya atribut kelas yang digunakan sebagai sisi kanan rule, sebagai contoh, $X \rightarrow Y$, Y harus merupakan atribut kelas [7].

Misalkan *dataset* training D mempunyai m atribut/item berbeda, $I = \{i_1, i_2, \ldots, i_m\}$ dan C adalah daftar dari label kelas. Banyaknya baris dalam D dinotasikan sebagai $|D|$. Atribut dapat bertipe kategorik atau kontinu. Pada kasus atribut bertipe kategorik, semua kemungkinan nilai dipetakan ke himpunan bilangan integer positif. Untuk atribut yang kontinu, atribut ini harus dideskritisasi untuk diubah ke bentuk kategorik.

Dalam masalah *associative classification*, setiap transaksi mengandung *subset* dari item-item yang dipilih dari I (d_i) dan label kelas c. Kumpulan dari nol atau lebih item disebut *itemset*. *Itemset* yang terdiri dari k item disebut *k-itemset*. Sebuah transaksi dikatakan mengandung *itemset* X jika X adalah *subset* dari d_j ($X \subseteq d_i$).

Rule $X \rightarrow c$ pada D mempunyai *condition support* α jika α% kasus pada D mengandung X. Rule $X \rightarrow c$ mempunyai *rule support* β jika β% kasus mengandung X dan dilabelkan dengan kelas c. Selanjutnya, rule $X \rightarrow c$ mempunyai nilai *confidence* δ jika δ% kasus pada D yang mengandung X dilabelkan dengan kelas c.

Itemset disebut *frequent itemset* jika nilai support dari *itemset* tersebut melampaui nilai *minimum support* (*minsupp*) *threshold*. Rule r disebut *frequent rule* jika nilai rule support dari r melewati nilai *minsupp threshold*. Sebuah rule diekstrak sebagai *Class Association Rules* (CAR) jika nilai *support* dan *confidence* dari r melewati nilai *minsupp* and *minimum confidence* (*minconf*) *threshold*.

2.2 Metode *Apriori*

Metode AC pertama yang menggunakan algoritma *Apriori* adalah metode CBA [7]. Prinsip *Apriori* menyatakan bahwa jika sebuah *itemset* adalah *frequent*, maka semua *subset*nya juga *frequent* [1].

Misalkan *Fk* menotasikan himpunan *frequent k-itemset*. Misalkan juga *Ck* adalah himpunan *k-itemset*. Algoritma *CAR generation* dari metode CBA dapat dilihat pada Gambar 1.

```
1   F1 = {large 1-itemset};
2   CAR1 = genRules(F1);
3   prCAR1 = pruneRules(CAR1);
4   for (k = 2; Fk-1 ≠ ∅; k++) do
5       Ck = candidateGen(Fk-1);
6       for each data case d ∈ D do
7           Cd = ruleSubset(Ck, d);
8           for each candidate c ∈ Cd do
9               c.condsupCount++;
10              if d.class = c.class then c.rulesupCount++
11          end
12      end
13      Fk = {c ∈ Ck | c.rulesupCount ≥ minsupp};
14      CARk = genRules(Fk);
15      prCARk = pruneRules(CARk);
16  end
17  CARs = Uk CARk;
18  prCARs = Uk prCARk;
```

Gambar 1. Algoritma *Rule Generation* CBA.

Sebagai ilustrasi dari algoritma CBA, perhatikan contoh *dataset* training pada Tabel 1. Misalkan diberikan nilai *minsupp threshold* 25% dan nilai *minconf threshold* 25%.

Tabel 1. Training Dataset

TID	Set of Items	Class Label
1	a, b, c	B
2	a, b, c, d	B
3	a, b, c	B
4	a, b, d	B
5	c, d	A
6	b, c	A
7	a, b, c	B
8	a, b, c	B

Pertama-tama *dataset* discan untuk memperoleh himpunan *frequent 1-itemset* (*F1*). Setelah itu nilai *confidence* dari *F1* dihitung. *F1* yang mempunyai nilai confidence lebih besar atau sama dengan nilai *minconf threshold* dibangkitkan sebagai *CAR 1-itemset*. Item yang dibangkitkan sebagai F1 adalah (a, B), (B, B), (c, B), (c, A), dan (d, B).

Selanjutnya, F1 digunakan untuk membangkitkan kandidat 2-*itemset* (*C2*), dengan menggabungkan dua elemen *F1*. *Dataset* discan lagi untuk memeriksa apakah nilai *rule support* dari *C2* memenuhi nilai *minsupp threshold*. Jika memenuhi, maka *C2* akan dibangkitkan sebagai frequent *2-itemset* (*F2*). *F2* yang mempunyai nilai *confidence* memenuhi *minconf threshold* akan diekstrak sebagai *CAR 2-itemset*.

Metode AC yang lain adalah metode MsCBA [8]. MsCBA memperbaiki algoritma *Apriori* dengan menggunakan nilai minsupp threshold lebih dari satu (*multiple class minimum support*, *minsups*). Metode ini menentukan *minsups* dengan memberikan nilai *minsupp threshold* yang berbeda pada setiap kelas tergantung dari frekwensi kelas tersebut. Misalkan $minsupp_i$ adalah nilai *minsupp* untuk kelas c_i dan sebuah nilai *global minsupp*, *g_minsupp* diberikan oleh user, maka: $minsupp_i = g\_minsupp * freqClass(c_i)$.

Metode LC adalah metode AC yang juga memperbaiki metode CBA [13]. Metode LC memperhatikan label kelas dari *frequent itemset* sebelum dilakukan penggabungan pada proses pembangkitan kandidat *itemset*. Hal ini berbeda dengan metode CBA yang tidak memperhatikan label kelas. Sebagai contoh, jika *A* dan *B* adalah dua *frequent itemset* yang pada iterasi k, maka LC akan menggabungkan *A* dan *B* menjadi kandidat *(k+1)-itemset* jika *A* dan *B* mempunyai label kelas yang sama.

2.3 Metode *FP-Growth Tree*

Metode *FP-Growth Tree* adalah metode untuk membangkitkan *frequent itemset* tanpa membangkitkan kandidat *itemset* seperti pada metode Apriori [5]. Metode ini menggunakan struktur data padat (*compact*) yang disebut *FP-tree* dan mengekstrak *frequent itemset* secara langsung dari struktur ini.

FP-tree dibentuk hanya dengan melakukan dua kali *scanning* pada *dataset*. *Scanning* yang pertama untuk memperoleh himpunan *frequent item* kemudian mengurutkan item sesuai nilai *support*nya dan *scanning* yang kedua untuk mengkonstruksi *FP-tree*.

FP-growth adalah algoritma yang membangkitkan *frequent itemset* dari *FP-tree* dengan menelusuri *FP-tree* secara *bottom-up*. Algoritma ini menggunakan strategi *divide-and-conquer* dalam memilah masalah menjadi lebih kecil dengan mengkonstruksi *conditional FP-tree* dari *global FP-tree* [5].

Salah satu teknik AC yang menggunakan algoritma *FP-growth tree* adalah metode CMAR [6]. CMAR menscan *dataset* training *D* untuk memperoleh *frequent 1-itemset* (*F*). Kemudian, CMAR mengurutkan item dalam *F* sesuai nilai *support*nya secara menurun, himpunan ini disebut *F-list*. Untuk setiap himpunan item (*set of items*, d_i) dalam *dataset* training, item yang muncul dalam *F-list* diekstrak dan diurutkan sesuai *F-list*.

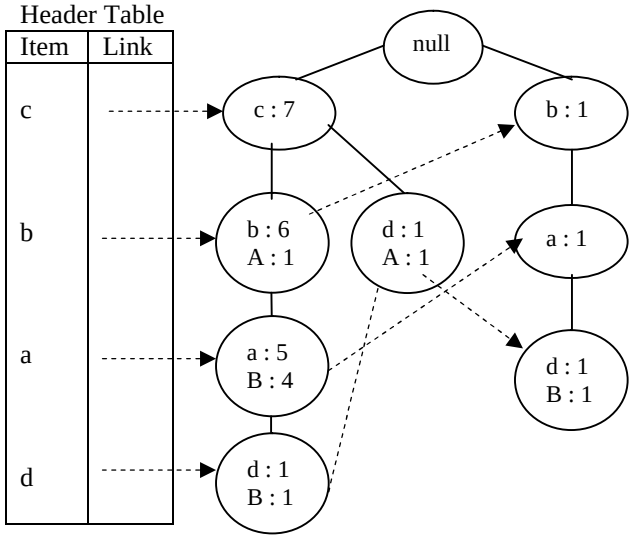
Tabel 2. Training Dataset sesuai *F-list*

TID	Set of Items	Class Label
1	c, b, a	B
2	c, b, a, d	B
3	c, b, a	B
4	b, a, d	B
5	c, d	A
6	c, b	A
7	c, b, a	B
8	c, b, a	B

Sebagai contoh, perhatikan *dataset* pada Tabel 1. Tabel 1 memperlihatkan *dataset* dengan 8 transaksi. Misal diberikan nilai *minsupp threshold* 25% dan nilai *minconf threshold* 25% Maka, *F-list* = $c - b - a - d$. Daftar *dataset* yang sesuai dengan *F-list* disajikan dalam Tabel 2.

Himpunan item yang pertama $\{c, b, a\}$ diekstrak dan disisipkan pada *FP-tree* sebagai cabang yang paling kiri. Label kelas dicantumkan pada *node* yang terakhir pada *path* tersebut.

Himpunan item yang kedua mempunyai prefiks yang sama dan dua item yang sama dengan dengan transaksi yang pertama. Sehingga, *path* untuk transaksi yang kedua *overlap* dengan *path* yang pertama. Frekwensi *node-node* yang bersesuaian dinaikkan. Hasil *FP-tree* bisa dilihat pada Gambar 2.



Gambar 2. The Global FP-tree

Berdasarkan *F-list*, CMAR membagi himpunan CAR kedalam 4 *subset* tanpa *overlap*: (1) himpunan yang mempunyai *d*, (2) mempunyai *a* tetapi tidak ada *d*, (3) mempunyai *b* tetapi tidak ada *a* atau *d*, dan (4) hanya mempunyai *c*.

Untuk menemukan *subset* rule yang mempunyai item *d*, CMAR menelusuri *path* yang mempunyai *node d* dan melihat "ke atas" untuk mengumpulkan *d-projected dataset*. Selanjutnya CMAR melakukan penambangan (*mining*) secara rekursif pada *conditional FP-tree*, dan memperoleh *itemset ad*, *bd*, *cd*, dan *bad*. Berdasarkan informasi distribusi label kelas, CMAR membangkitkan CAR: d→ B, ad → B, bd → B, dan BAD → B.

Metode lain yang menggunakan algoritma *FP-growth tree* adalah metode L3G [3]. Metode L3G mengkodekan rule yang lengkap menggunakan rule yang *compact* dengan memanfaatkan konsep *closed itemset* dan *essential rule* [2]. Algoritma *FP-growth tree diadopsi* untuk mengekstrak rule dari bentuk *compact* tersebut. Algoritma L3G dapat dilihat pada Gambar 3.

Sebagai ilustrasi dari algoritma L3G, perhatikan contoh *dataset* pada Tabel 1. Misal diberikan nilai *minsupp threshold* = 25% dan nilai *minconf threshold* = 25%. *Dataset* yang telah terurut disajikan pada Tabel 2. Pertama-tama *dataset* diproyeksikan terhadap item *d*, maka $G = d$ (lihat Gambar 4). Karena tidak ada item yang tercakup pada semua transaksi pada D_d, $T = \varnothing$. Kedua label kelas *B* dan *A* ada pada D_d, sehingga dua *compact rule* dapat diekstrak., $\{G = d, T = \varnothing\} \rightarrow B$ and $\{G = d, T = \varnothing\} \rightarrow A$. Masing-masing *compact rule* mengkodekan satu rule (lihat Gambar 4).

```
1   mine(D)
2.  {until all items in D are considered do
3       consider item i with the lowest rank
4.          {rec mine(projection(D, i), {i, ∅})
5           remove i from D}}

6   rec_mine(DG, {G, T})
7       {for each item i in DG | i is in all transactions in DG
8           {T = T ∪ i
9            remove i from DG}
10  for each class c in DG
11          check_rule_quality(DG, {G, T}  → c)

12  until all items in DG are considered do
13          consider item i with the lowest rank
14              {rec_mine(projection(DG, i), {G ∪ i, T})
15               remove i from DG}}
```

Gambar 3. Algoritma L3G Rule Extraction

D_d		
Tid	Items	Class Label
2	c, b, a	B
4	b, a	B
5	c	A

Compact Rule	Rule	Sup %	Conf %
{G=d, T=∅} → B	d → B	25	66.67
{G=d, T=∅} → A	d → A It is not CAR	12.5	33.33

Gambar 4. Projected Dataset D_d dan Compact Rule

D_d selanjutnya dapat diproyeksikan terhadap item *a* , b, dan c, sehingga menghasilkan projected *dataset* D_{da} ($G = da$), D_{db} ($G = db$), dan D_{dc} ($G = dc$). Proses yang dilakukan untuk menghasilkan *compact rule* sama seperti ilustrasi diatas.

Metode AC yang lain adalah metode DPPMine [4]. Metode ini memodifikasi algoritma *FP-growth tree* dengan mencari secara langsung *itemset* yang mempunyai nilai diskrimintif tinggi dengan menggunakan metode *branch-and-bound-search*. *Dataset* ditransformasikan ke bentuk *FP-tree* dan selanjutnya *itemset* yang diskriminatif dicari secara langsung.

Theori *upper bound* digunakan dalam metode pencarian *branch-and-bound* ini. Nilai *upper bound* yang digunakan pada metode ini adalah nilai *information gain*. Formula

information gain dan *information gain upper bound* disajikan pada [4].

Ide dasar dari metode DPPMine adalah, selama proses rekursi penambangan (*mining*) pada *FP-growth*, sebuah peubah global digunakan untuk menyimpan *itemset* yang paling diskriminatif yang telah ditemukan dan nilai *information gain*nya. Nilai *upper bound* dari *information gain* dari ukuran *conditional dataset* dihitung, sebelum mengkonstruksi *conditional FP-tree*.

```
1  if P = ∅
2      return;
3  for each item ai in P do
4      generate pattern β = ai ∪ α
       with support = ai.support;
5      compute information gain IG(β);
6      if IG(β) > maxIG
7          maxIG := IG(β);
8          bestP at := β;
9  construct pattern β's conditional database Dβ;
10 IGub(|Dβ|) := upper bound(|Dβ|);
11 if maxIG ≥ IGub(Dβ)
12     skip mining on Dβ;
13 else
14     construct β's conditional FP-tree Pβ;
15     branch and bound(Pβ, s, β);
```

Gambar 5. Algoritma Branch-and-Bound Mining

Gambar 5 memperlihatkan algoritma *branch-and-bound mining*. *IG*(*β*) pada baris 6 adalah *information gain* dari *frequent itemset β* dan *IGub*(|*Dβ*|) pada baris 10 adalah *information gain upper bound* yang diberikan oleh *conditional dataset Dβ*.

Untuk mengilustrasikan metode ini, perhatikan contoh pada Tabel 2. Misal diberikan nilai *minsupp threshold* 25% . *Global FP-tree* diperlihatkan pada Gambar 2.

Frequent itemset yang pertama dibangkitkan adalah item *d* dengan nilai *IG*(*d*) = 0.016. Maka *maxIG* adalah 0.016. *Conditional dataset* D_d disajikan pada Gambar 4 dengan ukuran *conditional dataset* adalah 3, nilai *information gain upper bound*nya adalah $IG_{ub}(3)$ = 0.467. Karena $IG_{ub}(3)$ > *maxIG*, *conditional FP-tree* terhadap *d* akan diekstrak untuk memperoleh *itemset* yang diskriminatif. *Mining* secara rekursif pada *conditional FP-tree* menghasilkan *itemset ad, bd, cd* dan *bad*, dengan *IG*(*ad*) = 0.123, *IG*(*bd*) = 0.123, *IG*(*cd*) = 0.074 and *IG*(*bad*) = 0.123.

Dengan cara yang sama seperti ilustrasi diatas diperoleh bahwa penambangan (*mining*) rule pada *conditional FP-tree* terhadap *a* tidak perlu dilakukan (bisa dilewatkan).

2.4 Metode *Tid-List Intersection*

Secara umum, terdapat dua representasi *dataset* yaitu layout horisontal [1] and vertikal [14]. Tabel 3 adalah contoh layout horisontal. Dalam pencarian *frequent itemsets* pada layout horisontal *dataset* discan secara berulang. *Scanning* dilakukan untuk menghitung nilai *support* dan melakukan pencocokan *itemset*.

Pada layout vertikal, *dataset* memuat sebuah group item dimana masing-masing item diikuti oleh tid-list [9], seperti terlihat pada Tabel 4, yang merupakan representasi vertikal dari Tabel 3. *Tid* dari sebuah *itemset* adalah lokasi transaksi pada *dataset* yang mengandung *itemset* tersebut.

Tabel 3. Training Dataset

TID	Set of Items	Class Label
1	a, b, e	A
2	b, c, d	B
3	c, e	B
4	c, d, e	B
5	a, b, c, d, f	A
6	c, e	B
7	a, b	A
8	a, b, c	A

Tabel 4. Representasi Vertikal dari Dataset

Itemset					Class Label	
a	b	c	d	e	A	B
1	1	2	2	1	1	2
5	2	3	4	3	5	3
7	5	4	5	4	7	4
8	7	5		6	8	6
	8	6				
		8				

Salah satu metode AC yang menggunakan algoritma *tid-list intersection* adalah metode MCAR [11]. Metode ini menscan satu kali *dataset* training untuk memperoleh *frequent item*. *Frequent item* ini akan disimpan dengan lokasi transaksinya (*tid*) pada sebuah array. *Frequent k-itemset* diperoleh dengan mengiriskan *tid-list* dari *frequent itemset* yang sudah ditemukan. *Tid-list* dari *frequent single itemset* juga digunakan untuk

menghitung nilai *support* and *confidence* dari rule *k-itemset*.

Sebagai ilustrasi dari metode CMAR, perhatikan contoh *dataset* pada Tabel 3. Misal diberikan nilai *minsupp threshold* 25% dan nilai *minconf threshold* 50%. Setelah scanning *dataset*, diperoleh bahwa nilai support dari item *f* tidak memenuhi nilai *minsupp threshold*, maka item *f* dihilangkan. Item yang lain beserta *tid*-nya disimpan secara vertikal seperti terlihat pada Tabel 4.

Tid-list dua item *F1* diiriskan untuk membangkitkan *frequent 2-itemset* (*F2*). Contoh, perhatikan item *a* dan *b* pada Tabel 4. Hasil irisan dari *tid-list*nya adalah {1, 5, 7, 8}. Ini adalah representasi dari *tid-list itemset ab*. Maka nilai *condition support itemset ab* adalah 4/8 * 100% = 50%.

Tid-list itemset ab dapat digunakan untuk menghitung nilai *rule support* dengan mengiriskan *tid-list itemset ab* dan setiap *tid-list* label kelas. Hasil pengirisan antara *tid-list itemset ab* dan label kelas *A* adalah {1, 5, 7, 8} dan label kelas *B* adalah ∅. Sehingga, tidak ada rule *ab* → *B* dan terdapat empat rule *ab* → *A*. Nilai rule support *ab* → *A* adalah 4/8 * 100% = 50% dengan nilai *confidence* 4/4 * 100% = 100%. Rule ini dibangkitkan sebagai *F2* dan sebagai CAR *2-itemset*. Kandidat *frequent 2-itemset* yang lain (*itemset ac*, *ad*, *ae*, *bc*, *bd*, *be*, *cd*, *ce*, dan *de*) dapat dibangkitkan dengan cara yang sama.

Metode CMAR diperbaiki dengan diusulkannya metode baru yang disebut metode CACA [10]. CACA memodifikasi algoritma *tid-list intersection* dengan mengecilkan jumlah kombinasi item yang diperlukan. Ide dasar dari metode CACA adalah membagi himpunan item ke dalam *k* himpunan yang lebih kecil sesuai label kelas.

Perhatikan item *a* dan label kelas *A*. hasil irisan antara *tid-list* mereka adalah {1, 5, 7, 8}. Nilai *rule support a* → *A* memenuhi nilai *minsupp threshold*. Item *a* dimasukkan ke himpunan *itemset* label kelas *A*, dinotasikan *C1*. Perhatikan item *c* dan label kelas *B*. Hasil irisan dari *tid-list* mereka adalah {2, 3, 4, 6}. Nilai *rule support c* → *B* memenuhi nilai *minsupp threshold*. Item *c* dimasukkan ke himpunan *itemset* label kelas *B*, dinotasikan *C2*.

Irisan antara setiap *tid-list* item dan setiap label kelas menghasilkan dua himpunan *itemset* *C1* = {*a*, *b*, *c*} dan *C2* = {*c*, *d*, *e*}. Element *C1* dan *C2* adalah *frequent 1-itemsets* (*F1*).

Frequent 2-itemset atau lebih dapat dilakukan dengan memperhatikan kombinasi dari elemen *C1* (untuk rule dengan label kelas *A*) dan kombinasi elemen *C2* (untuk rule dengan label kelas B). Perhatikan himpunan *C1* = {*a*, *b*, *c*}, maka rule yang mungkin adalah *ab* → *A*, *ac* → *A*, dan *bc* → *A*. Dari himpunan *C2* = {*c*, *d*, *e*}, rule yang mungkin adalah *cd* → *B*, *ce* → *B*, dan *de* → *B*. Nlai rule support dapat dihitung hanya dengan mengiriskan antara *tid-list 2-itemset* dan *tid-list* label kelasnya (contoh, antara *tid-list ab* dan label kelas *A*).

3. HASIL DAN PEMBAHASAN

Masalah yang paling penting pada fase *rule generation* adalah metode dalam menemukan *frequent itemset*. Algoritma Apriori adalah algoritma yang sering digunakan pada metode *associative classification* untuk menemukan *frequent itemset*. Algoritma ini membangkitkan kandidat *(k+1)-itemset* dengan menggabungkan dua *frequent k-itemset*. Semua *subset* dari sebuah *itemset* harus dibangkitkan, walaupun tidak semua *frequent itemset* berguna. Hal ini menyebabkan terdapat dua masalah utama pada algoritma ini: (1) pembangkitan kandidat *itemset* dalam jumlah yang sangat besar, dan (2) terjadi pengulangan scanning ke *dataset* dan pengecekan kandidat dengan pencocokan *itemset*.

Masalah ini diperbaiki oleh Han, et. al., (2004) dengan mengajukan algoritma baru yang disebut metode *FP-growth*. *FP-growth* membangkitkan *frequent itemset* tanpa melakukan pembangkitan kandidat *itemset*. Pada metode ini, stuktur data yang disebut FP-tree digunakan untuk menyimpan informasi frekwensi dari *itemset* dalam bentuk yang dimampatkan (*compact rule*). Metode ini hanya memerlukan dua kali scanning ke *dataset*, sehingga metode FP-growth jauh lebih cepat daripada metode Apriori.

Masalah yang masih dihadapi pada metode FP-*growth* adalah jumlah *frequent itemset* dan rule yang dibangkitkan masih cukup besar. Salah satu teknik yang digunakan untuk mengurangi jumlah *frequent itemset* adalah dengan memanfaatkan konsep nilai *information gain* dari *itemset* sehingga hanya *itemset* yang diskriminatif yang dibangkitkan [4].

Jumlah rule yang dibangkitkan dapat dikurangi dengan memampatkan bentuk rule (*compact rule*). Pemampatan bentuk rule, dapat dilakukan dengan menggunakan konsep *essential*

rule dengan memanfaatkan konsep *closed frequent itemset*.

Teknik lain yang digunakan untuk memperbaiki metode dalam membangkitkan *frequent itemset* adalah dengan memanfaatkan representasi *dataset* secara vertikal. Keuntungan utama pada teknik ini adalah nilai *support* dari *itemset* dihitung dengan mengiriskan *tid-list* dari *itemset* [9, 14]. Metode ini hanya memerlukan satu kali *scanning* ke *dataset* untuk menentukan *frequent 1-itemset*. Masalah yang dihadapi pada metode ini adalah ukuran awal dari *tid-list* kemungkinan terlalu besar agar bisa dimuat pada memori utama. Sehingga memerlukan teknik yang efektif untuk memampatkan ukuran *tid-list*.

Masalah lain dalam pembangkitan *class association rule* adalah penentuan nilai *minimum support threshold*. Kebanyakan teknik AC hanya menggunakan satu nilai *minimum support threshold* yang ditentukan oleh user. Cara penentuan nilai *minsupp threshold* ini kurang efektif terutama pada *dataset* transaksi dengan frekwensi kelas yang tidak seimbang. Teknik yang digunakan untuk memperbaiki kondisi ini adalah menentukan nilai *minsupp threshold* lebih dari satu dengan memperhatikan frekwensi label kelas dari data transaksi [8].

4. KESIMPULAN

Masalah utama yang dihadapi dalam membangkitkan *class association rule* pada metode *associative classification* adalah bagaimana membentuk algoritma yang efektif untuk menemukan *frequent itemset*. Beberapa teknik yang diusulkan oleh peneliti difokuskan pada pengurangan jumlah *frequent itemset* dan rule yang dibangkitkan. *Frequent itemset* set yang dibangkitkan seharusnya yang berguna (diskriminatif). Disamping itu, jumlah rule yang sangat besar seharusnya bisa dimampatkan sehingga tidak terlalu banyak menghabiskan ruang penyimpanan.

Fokus lain dari para peneliti adalah teknik dalam menentukan nilai *minimum support threshold*. Penentuan nilai *minsupp threshold* sangat mempengaruhi keakuratan rule yang dihasilkan.

Teknik lain pada metode AC terutama pada fase *rule generation* adalah pemanfaatan layout vertikal dari representasi *dataset*. Teknik ini sangat efektif karena hanya memerlukan satu kali *scanning* ke *dataset* transaksi dibandingkan dengan metode *Apriori* yang memerlukan perulangan *scanning* ke *dataset*.

5. SARAN

Teknik yang cukup menarik pada fase *rule generation* dari metode *associative classification* adalah pemanfaatan layout vertikal dari representasi dataset. Teknik ini sangat efektif karena hanya memerlukan satu kali scanning ke dataset transaksi. Salah satu kendala yang dihadapi pada pemanfaatan layout vertikal dari dataset adalah ukuran tid-list dari itemset yang sangat besar. Untuk mengatasi masalah ini, diperlukan pengkajian lebih lanjut tentang metode yang efektif untuk memampatkan ukuran tid-list dari itemset.

DAFTAR PUSTAKA

[1] Agrawal, R. & Srikant, R., 1994, Fast Algorithms for Mining Association Rule, *In Proceedings of the 20th International Conference on Very Large Data Base, Morgan Kaufmann*, Santiago, Chile, September 12-15.

[2] Baralis, E. & Chiosano, S., 2004, Essential Classification Rule Sets, *Journal of ACM Transactions on Database Systems*, vol. 29, no. 4, pp. 635–674

[3] Baralis, E., Chiusano, S. & Graza, P. 2004b, *On* Support Thresholds in Associative Classification, *In Proceedings of the 2004 ACM Symposium on Applied Computin*, Nicosia, Cyprus, March 14 – 17.

[4] Cheng, H., Yan, X., Han, J. & Yu, P.S., 2008, Direct Discriminative Pattern Mining for Effective Classification, *In Proceedings of the 24th IEEE International Conference on Data Engineering*, Cancún, México, April 7-12.

[5] Han, J., Pei, J., Yin, Y. & Mao, R., 2004, Mining Frequent Patterns without Candidate Generation: A Frequent Pattern Tree Approach, *Journal of Data Mining and Knowledge Discovery*, vol. 8, pp. 53-87

[6] Li, W., Han, J. & Pei, J., 2001, CMAR: Accurate and Efficient Classification Based on Multiple-Class Association Rule, *In Proceedings of the International Conference on Data Mining* (ICDM'01), San Jose, CA, November 29 – December 2.

[7] Liu, B., Hsu, W. & Ma, Y., 1998, Integrating Classification and Association Rule Mining, *In Proceedings of the International Conference on Knowledge Discovery and Data Mining*, New York, August 27 – 31.

[8] Liu, B., Ma, Y. & Wong, C.K., 2000, Improving an Association Rule Based Classifier, *In Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, Lyon, France, September 13-16.

[9] Savasere, A., Omiecinski, E. & Navathe, S., 1995, An Efficient Algorithm for Mining Association Rules in Large Databases, *In Proceedings of the 21st conference on Very Large Databases* (VLDB'95), Zurich,Switzerland, September 11-15.

[10] Tang, Z. & Liao, Q., 2007, A New Class Based Associative Classification Algorithm, *IAENG International Journal of Applied Mathematics*, vol. 36: 2.

[11] Thabtah, F., Cowling, P. & Peng, Y., 2005, MCAR: Multi-Class Classification Based on Association Rule Approach, *In Proceeding of the 3rd IEEE International Conference on Computer Systems and Applications*, Cairo, Egypt, January 3-6.

[12] Thabtah, F., 2007, A Review of Associative Classification Mining, *Journal of The Knowledge Engineering Review*, vol. 22:1, pp. 37-65.

[13] Thabtah, F., Mahmood, Q. & McCluskey, L., 2008, Looking at the Class Associative Classification Training Algorithm, *In Proceedings of the 5th International conference on Information Technology: New Generation*, Las Vegas, Nevada April 7-9.

[14] Zaki, M., Parthasarathy, S., Ogihara, M. & Li, W., 1997, New Algorithms for Fast Discovery of Association Rules, *In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, Newport Beach, CA, August 14-17.