PENINGKATAN PERFORMA ALGORITMA APRIORI UNTUK ATURAN ASOSIASI DATA MINING

Andreas Chandra

Teknik Informatika STMIK AMIKOM Yogyakarta
Jl Ring road Utara, Condongcatur, Sleman, Yogyakarta 55281
Email : andreaschaandra@yahoo.com

Abstrak

Aturan asosisasi memiliki beberapa algoritma. Salah satu yang popular adalah apriori. Apriori biasa digunakan untuk menemukan itemset yang sering muncul dan saling berkaitan dengan satu sama lain dalam suatu dataset. Dataset yang besar maka membutuhkan proses waktu yang lama. Penelitian ini membahas tentang cara meningkatkan pemrosesan data dalam algoritma apriori. Berdasarkan algoritma tersebtut, penelitian ini menunjukan batasan dari algoritma apriori asli yang sangat membuang banyak waku karena memperoses data yang ada di dataset keseluruhan untuk menemukan itemset yang sering muncul. Pada penelitian ini focus kepada menemukan cara untuk peningkatan performa algoritma yang baru dengan mengurangi pemindaian data. Pada percobaan ini menghasilkan dengan beberapa jumlah dataset dan beberapa minimum support yang dipakai pada algoritma apriori asli dan algoritma yang dikembangkan.

Kata kunci: *asosiasi, apriori, market basket analysis, data mining*

1. Pendahuluan

Pada era yang sangat modern ini sudah banyak toko-toko besar yang sudah menggunakan *E-commerce* dalam melakukan pemasaran, dengan menjajakan barang dagangan ke situs yang dimiliki. Pembeli dengan mudah menemukan barang yang dicari dan dapat mengetahui harga dan spesifikasi produk secara rinci. Dengan banyaknya toko-toko besar yang juga memiliki *E-commerce* untuk itu para pengelola harus mencermati pola-pola pembelian yang dilakukan oleh konsumen. Setiap transaksi yang terjadi akan selalu dicatat dan didokumentasikan. Pendokumentasian setiap transaksi sangat berguna bagi toko tersebut untuk segala keperluan. Data-data tersebut tersimpan dalam sebuah basis data berkapasitas besar. Bagi perusahaan atau supermarket, data-data yang tersimpan di basis data dapat dimanfaatkan untuk membuat laporan penjualan, kontrol inventaris, dan sebagainya. Data yang sangat banyak tersebut saat ini banyak perusahaan menggunakannya untuk keperluan anlisis data transaksi. Saat ini ada beberapa metode analisis data salah satunya dengan cara menemukan aturan asosiasi. Aturan asosiasi didapatkan dengan cara menggunakan algoritma apriori dimana untuk menemukan produk yang saling berhubungan. Saat ini perusahaan menggunakan metode pembangkitan itemset dengan cara memindai semua data

yang ada di dataset lalu membangkitkan semua item yang ada di dataset. Cara ini cukup mudah namun menimbulkan masalah baru. Dataset yang sangat besar membutuhkan proses yang sangat panjang. Proses yang sangat panjang maka membutuhkan memori yang sangat besar. Maka dari itu perlu sebuah algoritma untuk mempersingkat pemrosesan data yang efisien, agar data yang dihasilkan didapatkan dengan cepat. Dalam paper ini penulis merumuskan masalahnya adalah bagaimana membuat proses algoritma apriori menjadi lebih cepat untuk mendapatkan aturan asosiasi dan sebagai tujuan dari penelitiannya adalah membuat algoritma apriori menjadi lebih cepat dan diterpakan kedalam pembuatan aplikasi data mining.

Data Mining

Salah satu teknik yang dibuat dalam *data mining* adalah bagaimana menelusuri data yang ada untuk membangun sebuah model, kemudian menggunakan model tersebut agar dapat mengenali pola data yang lain yang tidak berada dalam basis data yang tersimpan[1]. Teknik utama dalamdata mining yaitu klasifikasi dan prediksi, pengelompokan, *outlier detection* aturan asosiasi, *sequence analysis*, time series analisis dan *text mining*[5].

Data Mining berisi pencarian trend atau pola yang diinginkan dalam *database* besar untuk membantu pengambilan keputusan di waktu yang akan dating. Pola-pola ini dkenali oleh perangkan tertentu yang dapat memberikan suatu analisa data yang berguna dan berwawasan yang kemudian dapat dipelajari dengan lebih teliti, yang mungkin saja menggunakan perangkat pendukung keputusan yang lainnya[2].

Proses *knowledge discovery in database* (KDD) secara garis besar dapat dijelaskan sebagai berikut: (1) *Data Selection*, Pemilihan (seleksi) data dari sekumpulan data operasional perlu dilakukan sebelum tahap penggalian informasi dalam KDD dimulai. Data hasil seleksi yang akan digunakan untuk proses *data mining*. Disimpan dalam suatu berkas, terpisah dari basis data operasional. (2) *Pre-processing / Cleaning*, sebelum proses *data mining* dapat dilaksanakan, perlu dilakukan proses *cleaning* pada data yang menjadi fokus KDD. Proses

cleaning mencakup antara lain membuat duplikasi data, memeriksa data yang inkonsisten, dan memperbaiki kesalahan pada data, seperti kesalahan cetak (tipografi).

(3) *Transformation, Coding* adalah proses transformasi pada data yang telah dipilih, sehingga data tersebut sesuai untuk proses *data mining*. Proses *coding* dalam KDD merupakan proses kreatif dan sangat tergantung pada jenis atau pola informasi yang akan dicar dalam basis data. (4) *Data Mining, data mining* adalah proses mencari pola atau informasi menarik dalam data terpilih dengan menggunakan teknik atau metode tertentu. Teknik, metode atau algoritma dalam *data mining* sangat bervariasi. Pemilihan metode atau algoritma yang tepat sangat bergantung pada tujuan dan proses KDD secara keseluruhan. (5) *Interpretation / Evaluation*, pola informasi yang dihasilkan dari proses data mining perlu ditampilkan dalam bentuk yang mudah dimengerti oleh pihak yang berkepentingan. Tahap ini merupakan bagian dari poses KDD yang disebut *interpretation*. Tahap ini mencakup pemeriksaan apakah pola atau informasi yang ditemukan bertentangan dengan fakta atau hipotesis yang ada sebelumnya[4].

2. Pembahasan

Pembahasan dalam paper ini lebih menjelaskan tentang algoritma apriori dan metode pembangkitan itemset selain itu dalam bagian ini juga akan dijelaskan mengenai peningkatan performa dari algoritma apiori beserta contohnya.
Sumber data yang digunakan berasal dari https://wiki.csc.calpoly.edu/datasets/attachment/wiki/apriori/apriori.zip dalam dataset ini hanya merepresentasikan bilangan angka 0-49.

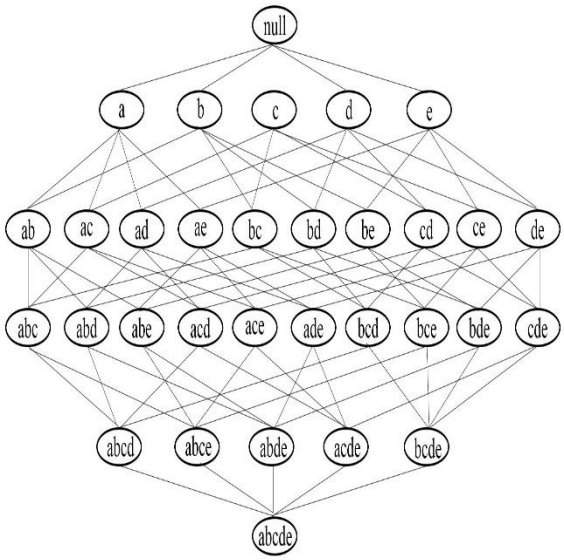
Algoritma Apriori

Algoritma apriori melakukan pembangkitan dan metodologi tes untuk menemukan itemset yang sering muncul, menghasilkan gabungan itemset yang lebih banyak dan secara berturut-turut yang sering muncul. Setiap ukuran yang berbeda dari kandidat itemset membutuhkan pemindaian dari dataset untuk menentukan apakah frekuensi kemunculannya memenuhi batas minimum[3].

Lebar transaksi didefinisikan sebagai jumlah item yang terdapat dalam sebuah transaksi. Suatu transaksi t_j dikatakna berisi sebuah itemset X jika X merupakan subset dari dari t_j[2].

Support count merupakan jumlah transaksi yang berisi suatu itemset tertentu atau dengan kata lain merupakan frekuensi kejadian dari suatu itemset. *Support* dari suatu itemset adalah perbandingan dari transaksi dalam basis data yang berisi semua itemset[2].

Dengan menggunakan metode *brute-force* item yang dibangkitkan dengan adanya 5 itemset maka terdapat 31 itemset yang dibangkitkan.



Gambar 1.*Pembangkitan Itemset*

Peningkatan Algoritma Apriori

Proses yang terjadi dalam algoritma apriori ketika memindai seluruh item yang ada di dataset, kemudian pemrosesan berlanjut untuk menghitung nilai *support count* untuk itemset pertama, itemset kedua, dan seterusnya, maka proses ini akan memakana waktu yang cukup lama. Peningkatan algoritma ini membutuhkan langkah langkah baru dalam pembangkitan itemset, meliputi:

1. jika dalam sebuah dataset terdapat transaksi T, dan I adalah item dari tiap pembelian, dan K-item adalah set dari sebuah item.
2. Andaikan σ(itemset) adalah support count atau frekuensi kemunculan itemset dalam sebuah transaksi yang terjadi.
3. Andaikan K_I adalah kandidat itemset yang akan dibangkitkan dari K-item.

Maka untuk pembangkitan itemset tersebut adalah memindai semua data yang ada untuk mendapatkan item yang ada di transaksi tersebut. Kemudian mengabungkan item item tersebut menjadi item. Lalu menghitung *support count* yang di dapatkan dari transaksi. Dalam pendekatan peneliti, peneliti meningkatkan proses algoritma apriori untuk mengurangi konsumsi waktu untuk pembangkitan calon itemset.

Peningkatan Algoritma Apriori

//Pembangkitan itemset dan penghitungan *support count*

(1) Set minsup.
(2) For(k=1 k<jumlah transaksi k++){

(3) C_k = generate itemset dari transaksi
(4) Foreach(datatransaksi){
(5) X = hitung C_k yang ada di transaksi
(6) }
(7) L_K = item X ≥ min_support
(8) End
(9) }

Analisis Aturan Asosiasi

Aturan asosiasi adalah pernyataan implikasi bentuk X → Y, di mana X dan Y adalah itemset yang lepas, yang memenuhi persamaan $X \cap Y = \{\}$. Kekuatan aturan asosiasi dapat diukur *support* dan *confidence*. *Support* digunakan untuk menentukan seberapa banyak aturan dapat diterapkan pada set data, sedangkan confidence digunakan untuk menentukan seberapa sering item di dalam Y muncul dalam transaksi yang berisi X.

Support, $s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}$ ……(1)

Confidence, $c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma N}$ ……...(2)

N adalah jumlah transaksi dalam set data.

Sifat *support* merupakan ukuran yang sangat penting dalam analisis asosiasi karena aturan yang sangat lemah nilai support-nya berarti asosiasi yang sangat jarang terjadi dalam set data[1].

Confidence digunakan untk mengukur keandalan dari inferensi yang dibuat oleh aturan. Untuk aturan X → Y, nilai *confidence* yang tinggi menandakan banyaknya Y yang muncul dalam transaksi X. *Confidence* juga memberikan cara untuk menemukan aturan asosiasi secara efisien[1].

Contoh Implementasi Peningkatan Algoritma Apriori

Misalkan kita memiliki dataset dengan 10 transaksi dan mengatur nilai minimum support = 3.

Tabel 1.*Tabel Transaksi*

Transaksi	Items
1	I_1, I_2, I_4
2	I_1, I_2, I_5
3	I_1, I_2
4	I_4, I_5
5	I_2, I_4, I_5
6	I_1, I_2
7	I_2, I_3, I_4
8	I_2, I_3, I_4
9	I_4, I_5, I_3
10	I_1, I_2, I_3

Kemudian membuat pembangkitan itemset

Tabel 2.*Kandidat 1-itemset*

Items	Support
I_1	5
I_2	8
I_3	4
I_4	6
I_5	4

Tahap selanjutnya adalah membangkitkan dengan 2 itemset dari pemabangkitan sebelumnya

Tabel 3.*Kandidat 2-itemset*

Kombinasi	Jumlah	
I_1,I_2	5	
I_1,I_3	1	Dihapus
I_1,I_4	1	Dihapus
I_1,I_5	1	Dihapus
I_2,I_3	3	
I_2,I_4	4	
I_2,I_5	2	Dihapus
I_3,I_4	3	
I_3,I_5	1	Dihapus
I_4,I_5	3	

Data yang digunakana untuk melakukan eksperimen ini didapat dari calpolyedu. Data data yang berisikan angka. https://wiki.csc.calpoly.edu/datasets/attachment/wiki/apriori/apriori.zip.
contoh data yang digunakan dalam pengujian ini ada di gambar 2.

```
4,5,6,10
1,12,19,46,47
16,32
18,24,35
12,22,27,30
18,29
0,2,8
14,40,44
3,18,35
12,28,31,36,48
14,44
6,27,28
17,29,47
4,16,32
0,2,4,7,11,37,42
13,40
14
13,20,32
12,31,36,48
6,42
18,20,22,24

22,41,47

24,28,35,42

5,19,22,25

9,21,29,30,42,45

18,35

14,44

16,32,45

27,28

6,14,22,30,38,40,46

23,24,40,41,43

1,19

5,22,37

0,2,5,46

4,9

27,28,33

5,20,22,41,44

14,34,39,44

1,12,16,22,39
2,9,16,17,35,45
```

Gambar 2.*Contoh Data*

karena akan terjadi pengurangan itemset pada peningkatan algoritma apriori, jika kita hitung jumlah itemset yang dibangkitkan dan algoritma apriori yang biasa maka jumlah itemset yang dibangkitkan akan berbeda jauh.

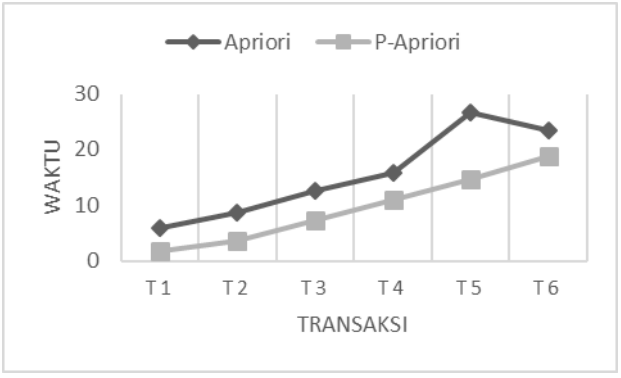
Tabel 4.*Jumlah Itemset yang dibangkitkan*

	Apriori	**P-Apriori**
1-itemset	50	50
2-itemset	1225	303
3-itemset	19600	743
jumlah	20875	1096

dalam memlakukan tes ini juga peneliti membagi jumlah row dalam dataset yang berbeda

- T1 = 500 transaksi
- T2 = 1000 transaksi.
- T3 = 2000 transaksi.
- T4 = 3000 transaksi
- T5 = 4000 transaksi.
- T6 = 5000 transaksi.

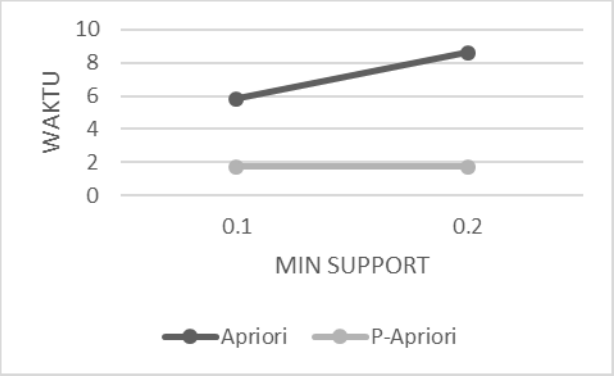
Uji coba yang pertama untuk membandingkan algoritma apriori dan algoritma peneliti adalah waktu konsumsi untuk mendapatkan aturan asosiasi, uji coba ini dilakukan dengan menggunakan 6 dataset dan untuk peningkatan algoritma apriorinya *minsupport* = 10. Hasil uji coba dapat dilihat di bawah ini.



Gambar 3.*Perbandingan konsumsi waktu*

Eksperimen kedua adalah dengan merubah nilai minimum support, pada percobaan pertama mengatur minimum support dengan nilai 50 (10%) dan yang kedua adalah 100(20%).

Dalam percobaan ini peneliti menggunakan dataset T1 yang berisi 500 transaksi. Percobaan ini juga untuk dapat melihat seberapa signifikan sebuah nilai minimum support mempengaruhi konsumsi waktu untuk mendapatkan aturan asosiasi.



Gambar 4.*Perbandingan minimum support*

3. Kesimpulan

Setiap proses data yang dilakukan melalui tahap *scanning*, kemudian menghitung *support count*, setelah itu dipilah mana support count yang memenuhi persyaratan sehingga, proses selanjutnya tidak memerlukan scanning pada file.

Daftar Pustaka

[1] E. Prasetyo, "Data Mining Konsep dan Aplikasi menggunakan MATLAB", Penerbit ANDI, 2012.
[2] F. A. Hermawati, "Data Mining", ANDI, 2013
[3] I. H. Witten, E. Frank, M. A. Hall, "Data Mining Practical Machine Learning Tools and Techniques 3rd Edition", 2011.
[4] Kusrini, E. T. Luthfi, "Algoritma Data Mining", Penerbit ANDI, 2009.
[5] Y. Zhao, "R and Data Mining: Example and Case Studies", Elsevier, 2012.

Biodata Penulis

Andreas Chandra, sedang menempuh pendidikan Strata 1, jurusan teknik informatika di STMIK AMIKOM Yogyakarta.

ISSN : 2302-3805