BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

2.1 Kajian Pustaka

Pada penelitian olah Mohammed J. Zaki, SPADE diterapkan pada dataset sintesis memberi hasil yang sangat baik, terutama jika dibandingkan dengan algoritma sequential pattern mining sebelumnya yaitu GSP [ZAK-01]. Pada peneletian oleh Zaki, untuk menentukan rule hanya menggunakan nilai confidence, sedangkan pada penelitian ini juga akan digunakan nilai lift rasio untuk lebih menguji kekuatan rule yang dibentuk.

2.2 Data Mining

2.2.1 Pengertian Data Mining

Pertumbuhan data pada transaksi yang pesat menyebabkan terciptanya kondisi yang bisa disebut sebagai kaya data tapi miskin informasi. Pada kondisi tersebut data – data hanya menjadi tumpukan data karena tidak dimanfaatkan. Untuk dapat memanfaatkan data – data tersebut dilakukan proses data mining. Data mining merupakan serangkaian proses untuk menggali nilai tambah dari suatu kumpulan data berupa pengetahuan yang selama ini tidak diketahui secara manual [PRA, 08].

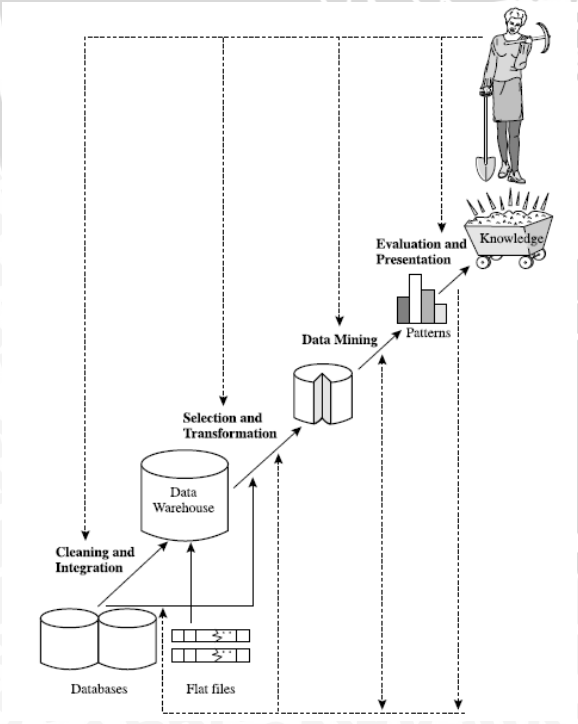
Kehadiran data mining dilatar belakangi dengan problema *data explosion* yang dialami oleh organisasi atau perusahaan selama beberapa tahun. Contoh suatu perusahaan *mini market* yang setiap harinya melakukan transaksi penjualan dan untuk setiap harinya *mini market* tersebut bisa melakukan rata – rata adalah 200 transaksi penjualan berarti dalam 1 tahun transaksi yang terjadi rata – rata adalah 73200 transaksi. Jika *mini market* tersebut memiliki beberapa outlet kemudian mini market tersebut sudah berjalan selama 4 tahun, dapat dibayangkan berapa banyak data yang sudah ada. Banyaknya data tersebut dapat dimanfaatkan untuk menggali informasi.

2.2.2 Tahap Data Mining

Data mining merupakan suatu rangkaian proses, data mining dapat dibagi menjadi beberapa proses:

a. *Data cleaning* : untuk menghilangkan *noise* data yang tidak konsisten.
b. *Data integration* : di mana sumber data yang terpecah dapat disatukan.
c. *Data selection* : di mana data yang relevan dengan tugas analisis dikembalikan ke dalam database.
d. *Data transformation* : di mana data berubah atau bersatu menjadi bentuk yang tepat untuk menambang dengan ringkasan performa atau operasi agresi.
e. *Data mining* : proses esensial di mana metode yang intelejen digunakan untuk mengekstrak pola data.
f. *Pattern evolution* : untuk mengidentifikasi pola yang benar – benar menarik yang mewakili pengetahuan berdasarkan atas beberapa tindakan yang menarik.
g. *Knowledge presentation* : di mana gambaran teknik visualisasi dan pengetahuan digunakan untuk memberikan pengetahuan yang telah ditambang kepada user.

Untuk lebih jelasnya dapat dilihat pada Gambar 2.1 [HAN, 06]:



Gambar 2.1 Proses *Knowledge discovery in databases*

2.3 Association Rule

Tugas utama dari *Association rule* adalah untuk mencari hubungan antar item, sebagai contoh seandainya pelanggan membeli sikat gigi maka biasanya akan membeli pasta gigi. Himpunan item disebut sebagai *itemset*. *Itemset* yang mengandung k items merupakan k-itemset. Kecenderungan kemunculan *itemset* dalam sejumlah transaksi disebut *frequency*, *support count* atau *count* atau *count* dari itemset.

Association rule mining ini digunakan untuk menemukan semua aturan *strong association rule* yang memenuhi *threshold minimum support* (min_sup) dan *threshold confidence minimum* (min_conf). Terdapat 2 langkah proses untuk menentukan *mining association rule* [EMH, 09], yaitu :

a. Menemukan *frequent itemset*. Berdasar definisi, masing – masing dari itemset akan muncul sedikitnya dengan *frequency* sebesar diberikan dalam *minimum support count*.
b. Munculkan *strong association rule* dari *frequent itemset*. Berdasar definisi, aturan ini harus memenuhi *minimum support* dan *minimum confidence*.

Pada prosesnya untuk mencari hubungan antar item tersebut akan dilakukan berulang – ulang untuk menentukan pola hubungan yang berbeda – beda. Proses yang berulang – ulang menyebabkan waktu yang dibutuhkan untuk menemukan hubungan antar item, sehingga dibutuhkan algoritma untuk menyelesaikannya.

2.3.1 Support dan Confidence

Support adalah ukuran yang menunjukkan besar tingkat dominasi suatu item atau *itemset* keseluruhan transaksi [HAN, 06]. *Support* digunakan untuk rule yang memiliki nilai yang kurang dari *threshold* untuk menentukan *support* adalah:

$$Support, S(A \rightarrow B) = \frac{\sigma(A \cup B)}{N} \quad (2.1)$$

Dimana :

- $\sigma(A \cup B)$ = Jumlah itemset di semua transaksi
- N = Jumlah total transaksi

Confidence adalah ukuran yang menunjukkan hubungan antar 2 item secara kondisional [HAN, 06]. Nilai *confidence* keandalan dari rule yang dibuat. Rumus yang digunakan untuk menentukan *confidence* adalah :

$$Confidence, C(A \rightarrow B) = \frac{\sigma(A \cup B)}{\sigma(A)} \quad (2.2)$$

Dimana :

- $\sigma(A \cup B)$ = Jumlah itemset di semua transaksi
- $\sigma(A)$ = Jumlah *antecedent* pada transaksi

2.4 Sequential Pattern Mining

Sequential pattern mining digunakan untuk mencari data yang memiliki urutan, data tersebut bisa merupakan urutan transaksi. *Sequential pattern mining* pertama kali diperkenalkan oleh Agrawal dan Srikant. Proses *sequential pattern mining* dapat digambarkan sebagai berikut, diberikan sejumlah *sequence*, setiap *sequence* terdiri atas sederetan elemen, dan setiap elemen terdiri atas sejumlah item, serta diberikan nilai *minimum-support*. Penggalian pola *sequential* adalah pencarian semua *subsequence* berulang, yaitu *subsequence* yang frekuensi kejadiannya lebih besar dari *minimum-support* [AGR, 95].

Untuk menyelesaikan permasalahan sekuensial ini dapat dilakukan dengan beberapa metode antara lain *Generalizes Sequential Pattern* atau disebut GSP, FreeSpan, dan PrefixSpan. Sebagai contoh proses *sequential pattern mining*, terdapat tabel transaksi penjualan yang berisikan *customer*, tanggal dan item. Dari tabel transaksi tersebut kemudian dibentuk *sequence* transaksi berdasarkan *customer* dan diurutkan berdasar tanggal sehingga membentuk beberapa *sequence*. Untuk lebih jelas, dapat dilihat contoh berikut [AGR, 95]:

Tabel 2.1: *Customer Transaction*

Customer ID	Transaction Time	Item
1	25-06-2001	30
	30-06-2001	90
2	13-06-2001	10,20
	15-06-2001	30
2	20-06-2001	40,60,70
3	25-06-2001	30,50,70
4	25-06-2001	30
	30-06-2001	40,70
	25-06-2001	90
5	13-06-2001	90

Tabel 2.2: *Customer Sequence*

Customer ID	Customer Sequence
1	{(30),(90)}
2	{(10,20),(30),(40,60,70)}
3	{(30,50,70)}
4	{(30),(40,70),(90)}
5	{(90)}

Tabel 2.1 adalah tabel transaksi dari *customer*, dari tabel tersebut dibentuk beberapa *sequence* pada Tabel 2.2. Setelah didapat *sequence* transaksi kemudian dicari *subsequence* yang lebih dari *minimum-support*, contoh kali ini menggunakan *minimum-support* sebesar 25%. Sehingga menghasilkan 2 *sequential* {(30) (90)} dan {(30) (40,70)}.

Tabel 2.3: *Sequence* yang lebih dari *min-sup*

Sequential Pattern With Support = 25%
{(30),(90)}
{(30),(40,70)}

Hasil *pattern* {(30) (90)} didapat dari *customer* 1 dan 4. Dalam *customer* 4 terdapat item (40,70) antara (30) dan (90) akan tetapi karena *pattern* {(30) (90)} maka (40,70) tidak perlu dimasukkan. Hasil untuk *pattern* {(30) (40,70)} didapat dari *customer* 2 dan 4. *Customer* 2 membeli 60 bersamaan membeli 40 dan 70, jadi *pattern* (40,70) adalah subset dari (40,60,70).

2.5 Lift Rasio

Lift rasio digunakan untuk mengukur seberapa kuat rule yang dibentuk dari algoritma SPADE. Nilai *lift* rasio berkisar antara 0 sampai dengan tak terhingga. Nilai minimum dari *lift* rasio tidak ditentukan seperti halnya *support* atau *confidence*. Jika nilai *lift* rasio kurang dari 1 dalam hal ini adalah nilai minimum maka *rule antecedent* berpengaruh negatif pada *rule consequent*. Jika nilai *lift* rasio sama dengan 1 maka rule tersebut sering muncul bersamaan tetapi independen. Rule yang independen merupakan rule dimana untuk mendapatkan *consequent* tidak tergantung pada *antecedent*. Pada *lift* rasio, rule yang direkomendasikan adalah jika *lift* rasio lebih dari 1 karena *antecedent* memiliki pengaruh positif pada *consequent*. Berikut rumus untuk menentukan *lift* rasio [FOM, 11]:

$$Expected\ Confidence, EC\ (A \rightarrow B) = \frac{\sigma(B)}{m} \quad (2.3)$$

$$Lift = \frac{Confidence}{Expected\ Confidence} \quad (2.4)$$

Dimana:

- $\sigma(B)$= Jumlah *consequent* dalam transaksi
- m = Jumlah transaksi

2.6 Algoritma SPADE (Sequential PAttern Discovery using Equivalent classes)

Sequential PAttern Discovery using Equivalent classes adalah salah satu algoritma *sequential pattern mining* yang menggunakan format data vertikal pada database *sequence*. Dalam format data vertikal, database sequence menjadi berbentuk kumpulan urutan yang formatnya [*itemset* :(*sequence_ID, eventID*)]. Dengan kata lain, untuk setiap itemset akan disimpan *sequence identifier* dan *event identifier* yang berkoresponden. *Event identifier* berguna sebagai *timestamp*

atau penanda waktu dari itemset tersebut. Sepasang (*sequence_ID, eventID*) untuk setiap itemset membentuk ID_list dari itemset tersebut.

Langkah – langkah algoritma SPADE dalam mencari frequent sequence kemudian menentukan rule dari frequent sequence tersebut adalah sebagai berikut:

1. Menghitung frequent 1-sequnce

Untuk mencari frequent 1-sequence dari sequence database yang harus dilakukan adalah dilakukan scan untuk setiap itemset dalam sequence database. Untuk masing – masing itemset kita simpan id-listnya (pasangan sid dan eid). Kemudian kita scan id-list dari masing – masing id-list tersebut, setiap ditemui sid yang sebelumnya belum ada maka nilai supportnya ditambah. Sequence yang dimasukkan dalam frequent 1-sequence adalah yang supportnya lebih dari min_sup.

2. Menghitung frequent 2-sequence

Dalam mencari frequent 2-sequence, data yang digunakan adalah data dari frequent 1-sequence, sehingga tidak perlu mencari dari sequence database lagi. Untuk setiap masing – masing frequent 1-sequence kita gabungkan dengan semua frequent 1-sequence lainnya. Contohnya jika 1-sequence A digabungkan dengan 1-sequence B maka kemungkinan 2 sequence yang terjadi adalah A,B dimana A dan B muncul bersamaan dalam transaksi, A → B dimana item B muncul setelah item A, dan B → A dimana item B muncul setelah item A. Untuk setiap masing – masing penggabungan frequent 1-sequence ini dilakukan pengecekan apakah dalam id-listnya memiliki sid yang sama, jika sama kemudian dilakukan pengecekan apakah eid dari 1-sequence A sama dengan, kurang dari atau lebih dari eid 1-sequence B. Apabila sama maka id-listnya dimasukkan dalam 2-sequence A,B. Jika eid B lebih besar dari A maka id-listnya dimasukkan dalam 2-sequence A → B dan jika eid A lebih besar dari B maka id-listnya dimasukkan dalam 2-sequence B → A. Kemudian seperti dalam frequent 1-sequence kita tambahkan supportnya untuk setiap masing – masing sid yang sebelumnya belum ditemui. Dari 2-sequence itu kemudian dilakukan pengecekan apakah supportnya lebih dari min_sup. Jika memenuhi syarat maka dimasukkan dalam frequent 2-sequence.

3. Menentukan frequent k-sequence.

Setelah mencari frequent 2-sequence, untuk mencari frequent sequence – frequent sequence berikutnya dilakukan proses yang sama, yaitu mencari frequent k-sequence. Untuk mencari frequent k-sequence ini dilakukan join pada frequent (k-1) sequence yang memiliki prefix yang sama. Contohnya untuk mencari 3-sequence kita gabungkan frequent sequence dari 2-sequence yang memiliki prefix yang sama, untuk mencari 4-sequence kita gabungkan frequent sequence dari 3-sequence yang memiliki prefix yang sama, dan seterusnya. Untuk mencari prefix frequent (k-1) sequence kita hilangkan item terakhir dari sequence tersebut. Contoh jika terdapat 4-sequence A → B → C → D, maka prefixnya adalah A → B → C. Untuk setiap penggabungan ini ada 3 kemungkinan hasil:

a. Jika A,B digabungkan dengan A,C, maka kemungkinan hasilnya hanya A,B,C.
b. Jika A,B digabungkan dengan A → C, maka kemungkinan hasilnya hanya A,B → C.
c. Jika A → B digabungkan dengan A → C, maka ada 3 kemungkinan hasil: A → B,C, dan A → B → C dan A → C → B.

Dari setiap kemungkinan ini kita cek supportnya apakah memenuhi min_sup, jika ya maka sequence itu termasuk dalam frequent k-sequence. Pencarian frequent sequence dihentikan apabila tidak ada frequent (k-1) sequence yang bisa dijoin atau sudah tidak ditemukan frequent k-sequence lagi.

4. Pembentukan Rule

Setelah ditemukan semua frequent sequence, ditentukan rule dari sequence – sequence tersebut. 1-sequence tidak digunakan untuk membentuk rule karena hanya terdiri dari 1 item. Untuk 2-sequence yang menjadi antecedent adalah item pertama dan consequentnya adalah item keduanya. Contoh untuk sequence A → B maka rule yang dibentuk adalah A => B. Sedangkan untuk sequence yang panjangnya lebih dari 2 atau k-sequence, yang dijadikan consequent adalah item terakhir, sedangkan antecedentnya adalah semua item sebelum item terakhir. Contohnya pada 4-sequence A → B → C → D, maka rule yang dihasilkan adalah A → B → C => D. Untuk masing – masing rule dihitung nilai confidencenya menggunakan rumus 2.2. Jika rule tersebut memenuhi batas min_conf, maka rule

itu diterima. Kemudian dari rule yang diterima tersebut kita hitung nilai lift rasionya menggunakan rumus 2.4. Nilai lift rasio semakin besar semakin baik, dengan batas 1. Apabila rule memiliki nilai lift lebih dari atau sama dengan 1, maka dalam rule itu antecedent memiliki pengaruh positif terhadap consequent. Sehingga rule dinyatakan baik, sebaliknya apabila nilai lift kurang dari 1 maka rule dianggap kurang baik.