Minería de Datos
IIC2433

Reglas de asociación

Vicente Domínguez

¿Qué veremos esta clase?

- Reglas de asociación

Tenemos un pequeño problema…

- Con 5 items, obtuvimos 2^5 - 1 posibles ítems
  - Es decir, 31

TID	Items
1	Pan, Coca Cola, Leche
2	Cerveza, Pan
3	Pan, Coca Cola, Pañales, Leche
4	Cerveza, Pan, Pañales, Leche
5	Coca Cola, Pañales, Leche

- {Pan}
- {Coca Cola}
- {Leche}
- {Pañales}
- {Cerveza}
- {Pan, Coca Cola}
- {Pan, Leche}
- {Pan, Pañales}
- {Pan, Cerveza}
- {Coca Cola, Leche}
- {Coca Cola, Pañales}
- {Coca Cola, Cerveza}
- {Leche, Pañales}
- {Leche, Cerveza}
- {Pañales, Cerveza}
- {Pan, Coca Cola, Leche}
- {Pan, Coca Cola, Pañales}
- {Pan, Coca Cola, Cerveza}
- {Pan, Leche, Pañales}
- {Pan, Leche, Cerveza}
- {Pan, Pañales, Cerveza}
- {Coca Cola, Leche, Pañales}
- {Coca Cola, Leche, Cerveza}
- {Coca Cola, Pañales, Cerveza}
- {Leche, Pañales, Cerveza}
- {Pan, Coca Cola, Leche, Pañales}
- {Pan, Coca Cola, Leche, Cerveza}
- {Pan, Coca Cola, Pañales, Cerveza}
- {Pan, Leche, Pañales, Cerveza}
- {Coca Cola, Leche, Pañales, Cerveza}
- {Pan, Coca Cola, Leche, Pañales, Cerveza}

Tenemos un pequeño problema…

- Con 5 items, obtuvimos 2^5 - 1 posibles ítems
  - Es decir, 31
- Con *n* items, tenemos 2^n - 1 posibles ítems
- Una tienda suele tener varios ítems
  - Imaginemos n = 100

Tenemos un pequeño problema…

- Si n = 100
- 2^{100} - 1 posibles ítems
- 2^{100} - 1 =

  1267650600228229401496703205376

Un computador actual puede hacer ~ 3 millones de operaciones por segundo (3 GHz)...

Entonces demoraríamos aproximadamente

64403322675823264816693248 segundos en sólo encontrar los itemsets posibles

es decir 204086005124396249 7 años

Solución: Algoritmo **Apriori**

Principio de Monotonicidad:

Si un itemset es frecuente, entonces todos los subgrupos de éste también son frecuentes

- Si *{pan, cerveza}* es frecuente, entonces *{pan}* y *{cerveza}* deben ser frecuentes.
- Si *{pan, cerveza, pañales},* entonces…

Regla inversa (anti-monotonía):

Si un itemset no es frecuente, entonces todos sus supersets deben también ser infrecuentes

- Si *{pan}* **no** es frecuente, entonces ningún conjunto que contenga panes será frecuente
- Si {pan, coca cola}, entonces...

Algoritmo **Apriori**

- **¿Es la solución perfecta?**
- **¿Tiene algún tipo de problema?**

Algoritmo **Apriori**

Algunos problemas

- **Consume mucha memoria**

- **El manejo de ítems como strings hace que el algoritmo sea más pesado**
- **Probar combinaciones a posibles candidatos puede ser muy lento**
- **Cada vez que se cuentan se itera sobre las transacciones para contar.**

Algoritmo **FP-Growth**

- Solución ideada para suplir los problemas de Apriori
- Se basa en una estructura de árbol llamada FP-Tree
- En el árbol, cada nodo representa un ítem y su contador de apariciones
- Una rama completa representa un itemset.

Algoritmo **FP-Growth**

Paso a paso

- El algoritmo se compone de dos grandes pasos

  a. Creación del FP-Tree
  b. Minar el FP-Tree

Creación **FP-Tree**

Primero, los datos

TID	Items
1	Pan, Coca Cola, Leche
2	Cerveza, Pan
3	Pan, Coca Cola, Leche
4	Cerveza, Pan, Pañales, Leche
5	Pan, Coca Cola, Leche

Creación **FP-Tree**

TID	Items
1	Pan, Coca Cola, Leche
2	Cerveza, Pan
3	Pan, Coca Cola, Leche
4	Cerveza, Pan, Pañales, Leche
5	Pan, Coca Cola, Leche

1. Calculamos el soporte de los *1-itemsets* (itemsets de tamaño 1)

- σ({Pan}) =
- σ({Coca Cola}) =
- σ({Leche}) =
- σ({Cerveza}) =
- σ({Pañales}) =

Creación **FP-Tree**

TID	Items
1	Pan, Coca Cola, Leche
2	Cerveza, Pan
3	Pan, Coca Cola, Leche
4	Cerveza, Pan, Pañales, Leche
5	Pan, Coca Cola, Leche

1. Calculamos el soporte de los *1-itemsets* (itemsets de tamaño 1)

- σ({Pan}) = 5
- σ({Coca Cola}) = 3
- σ({Leche}) = 4
- σ({Cerveza}) = 2
- σ({Pañales}) = 1

Creación **FP-Tree**

TID	Items
1	Pan, Coca Cola, Leche
2	Cerveza, Pan
3	Pan, Coca Cola, Leche
4	Cerveza, Pan, Pañales, Leche
5	Pan, Coca Cola, Leche

2. Definimos el valor del umbral y lo aplicamos

Umbral = 0,3

- σ({Pan}) = 5
- σ({Coca Cola}) = 3
- σ({Leche}) = 4
- σ({Cerveza}) = 2
- σ({Pañales}) = 1

Creación **FP-Tree**

TID	Items
1	Pan, Coca Cola, Leche
2	Cerveza, Pan
3	Pan, Coca Cola, Leche
4	Cerveza, Pan, Pañales, Leche
5	Pan, Coca Cola, Leche

2. Definimos el valor del umbral y lo aplicamos

Umbral = 0,3

- σ({Pan}) = 5
- σ({Coca Cola}) = 3
- σ({Leche}) = 4
- σ({Cerveza}) = 2
- **σ({Pañales}) = 1**

Creación **FP-Tree**

TID	Items
1	Pan, Coca Cola, Leche
2	Cerveza, Pan
3	Pan, Coca Cola, Leche
4	Cerveza, Pan, Pañales, Leche
5	Pan, Coca Cola, Leche

3. Ordenamos los datos según soporte

- σ({Pan}) = 5
- σ({Leche}) = 4
- σ({Coca Cola}) = 3
- σ({Cerveza}) = 2

Creación **FP-Tree**

4. Ir agregando las transacciones según orden de soporte

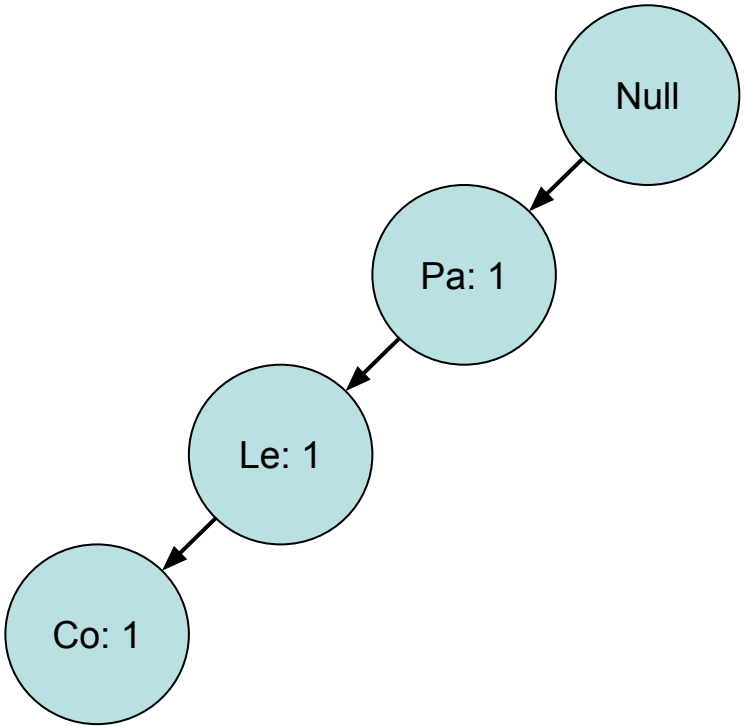
TID	Items
1	Pan, Coca Cola, Leche
2	Cerveza, Pan
3	Pan, Coca Cola, Leche
4	Cerveza, Pan, Pañales, Leche
5	Pan, Coca Cola, Leche

TID	Items
1	Pan, Leche, Coca Cola
2	Pan, Cerveza
3	Pan, Leche, Coca Cola
4	Pan, Leche, Cerveza
5	Pan, Leche, Coca Cola

Creación **FP-Tree**

4. Ir agregando las transacciones según orden de soporte

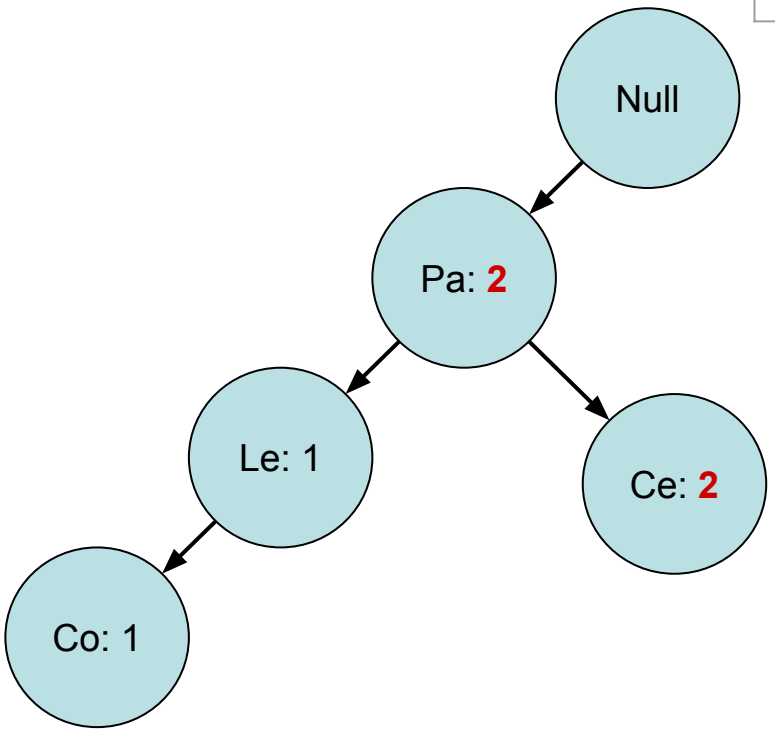
TID	Items
1	Pan, Leche, Coca Cola



Creación **FP-Tree**

4. Ir agregando las transacciones según orden de soporte

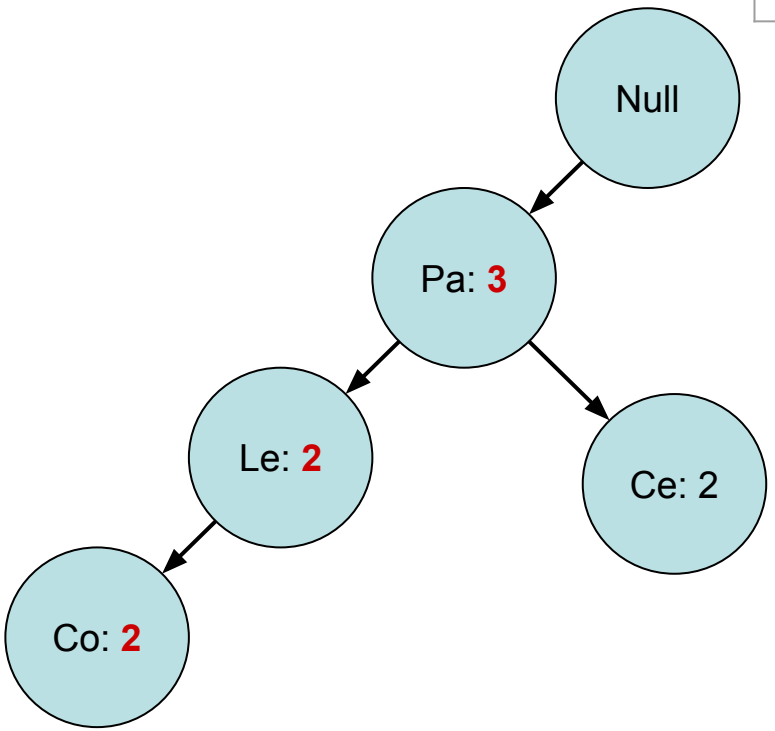
TID	Items
2	Pan, Cerveza



Creación **FP-Tree**

4. Ir agregando las transacciones según orden de soporte

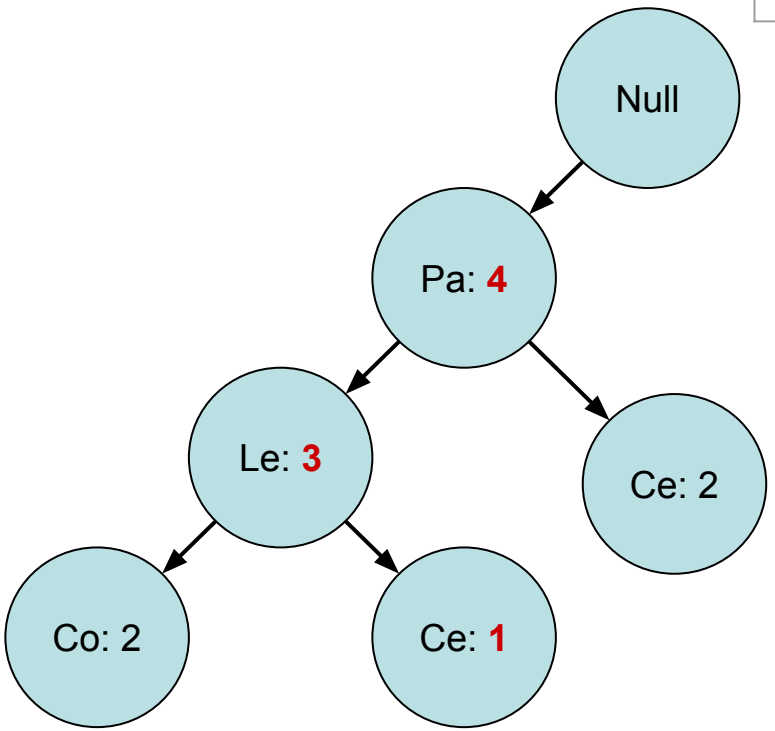
TID	Items
3	Pan, Leche, Coca Cola



Creación **FP-Tree**

4. Ir agregando las transacciones según orden de soporte

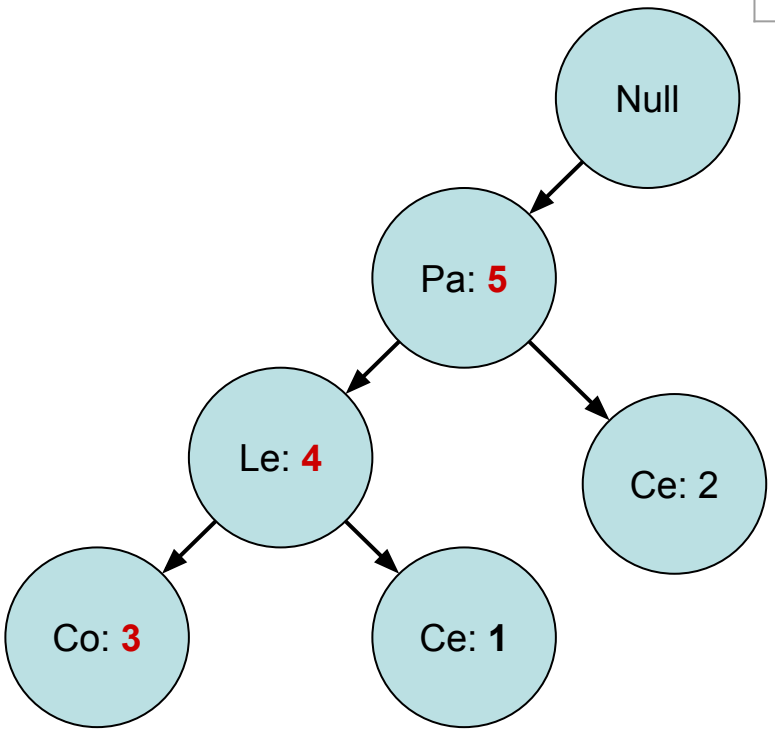
TID	Items
4	Pan, Leche, Cerveza



Creación **FP-Tree**

4. Ir agregando las transacciones según orden de soporte

TID	Items
5	Pan, Leche, Coca Cola



Minando el **FP-Tree**

1. Utilizamos los ítems de menor a mayor soporte

- σ({Cerveza}) = 2
- σ({Coca Cola}) = 3
- σ({Leche}) = 4
- σ({Pan}) = 5

2. Buscamos itemset frecuentes en los caminos que llegan a los items de menor frecuencia.

- Caminos que llegan a Cerveza
- Caminos que llegan a Coca Cola
- ...

Algoritmo **FP-Tree**

Buscando reglas de asociación

Para cada itemset frecuente sacar sus subconjuntos y calcular la **confianza** entre ellas.

Algoritmo **FP-Tree**

Beneficios

- Evita la generación de candidatos en cada iteración
- Pasa por el dataset completo a lo más 2 veces, por lo tanto es O(n)
- Reduce la cantidad de memoria utilizada para almacenar la base de datos

Ejercicio

TID	Items
T1	I1, I2, I5
T2	I2, I4
T3	I2, I3
T4	I1, I2, I4
T5	I1, I3
T6	I2, I3
T7	I1, I3
T8	I1, I2, I3, I5
T9	I1, I2, I3