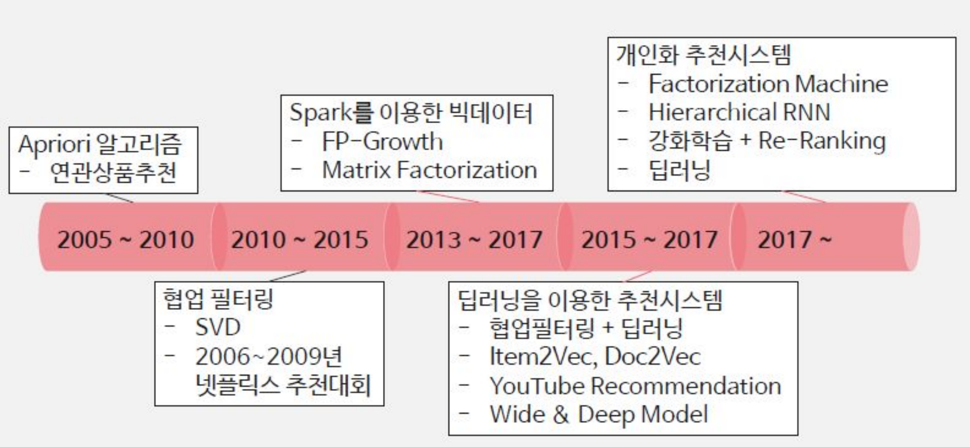
Association Analysis, Apriori, FP-Growth

Jiho Kang
2022.03.19

1. History of RecSys



2005 ~ 2010

Apriori 알고리즘
- 연관상품추천

2010 ~ 2015

협업 필터링
- SVD
- 2006~2009년 넷플릭스 추천대회

2013 ~ 2017

Spark를 이용한 빅데이터
- FP-Growth
- Matrix Factorization

2015 ~ 2017

딥러닝을 이용한 추천시스템
- 협업필터링 + 딥러닝
- Item2Vec, Doc2Vec
- YouTube Recommendation
- Wide & Deep Model

2017 ~

개인화 추천시스템
- Factorization Machine
- Hierarchical RNN
- 강화학습 + Re-Ranking
- 딥러닝

SKplanet Tacademy - 추천시스템 분석 입문하기 1강

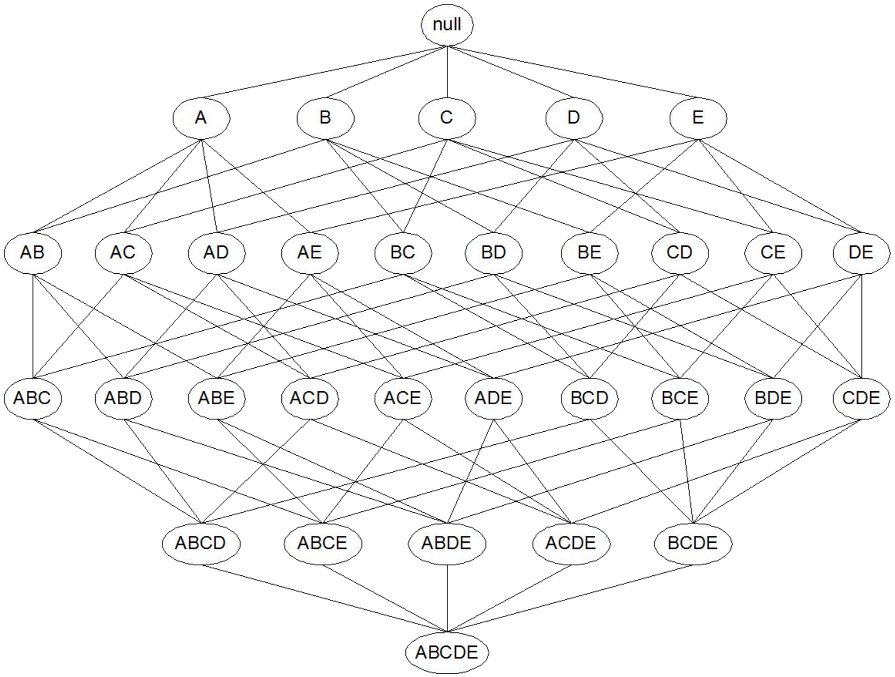
- We’re going to see the history of recommendation system.
- Starting from Apriori Algorithm, SVD, to Deep learning.

2. Association Analysis (연관분석)

- What is Association Analysis?
  - Algorithm to find out how the Item A relates to the Item B.
  - Rule-based model
  - Permutation

- How to find a relation?
  - How frequent are they purchased together?
  - Did the person who bought item A buy item B together?

- Disadvantage
  - As the number of items increases, the number of cases increases exponentially.
    ㄴ> *Apriori*

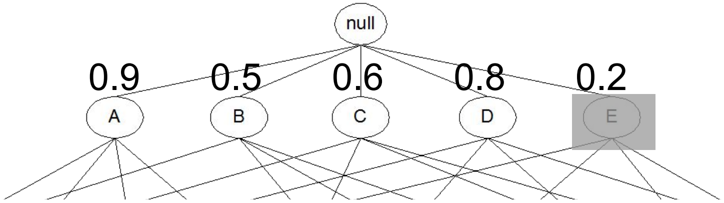


https://chih-ling-hsu.github.io

3. Apriori

- What is Apriori?
  - To reduce the number of item set in association analysis.
  - The case, which includes infrequent item, will also not be frequent.

- How does it work?
  1. Make one-item-set frequent set with items.
  2. Select cases above the minimum support.
  3. Make two-item-set with survived items.
  4. Select cases above the minimum support
  5. Repeat until k-item-set



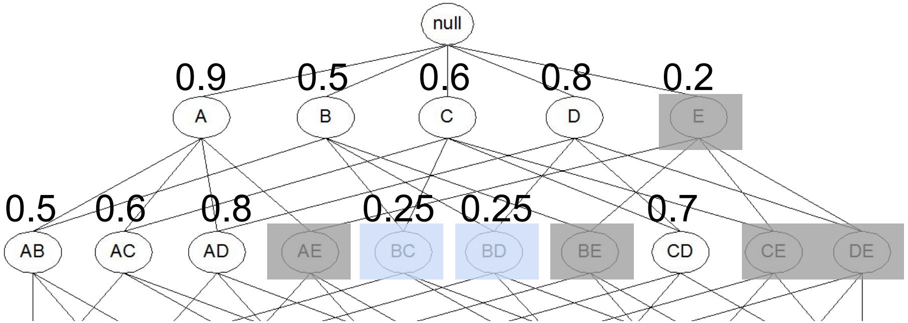
- Advantage
  - Algorithm is simple and comprehensible.

- Disadvantage
  - Runtime increases exponentially with the amount of item. => *FP-Growth*
  - It shows only correlations, not cause and effect.

3. Apriori

- What is Apriori?
  - To reduce the number of item set in association analysis.
  - The case, which includes infrequent item, will also not be frequent.

- How does it work?
  1. Make one-item-set frequent set with items.
  2. Select cases above the minimum support.
  3. Make two-item-set with survived items.
  4. Select cases above the minimum support
  5. Repeat until k-item-set



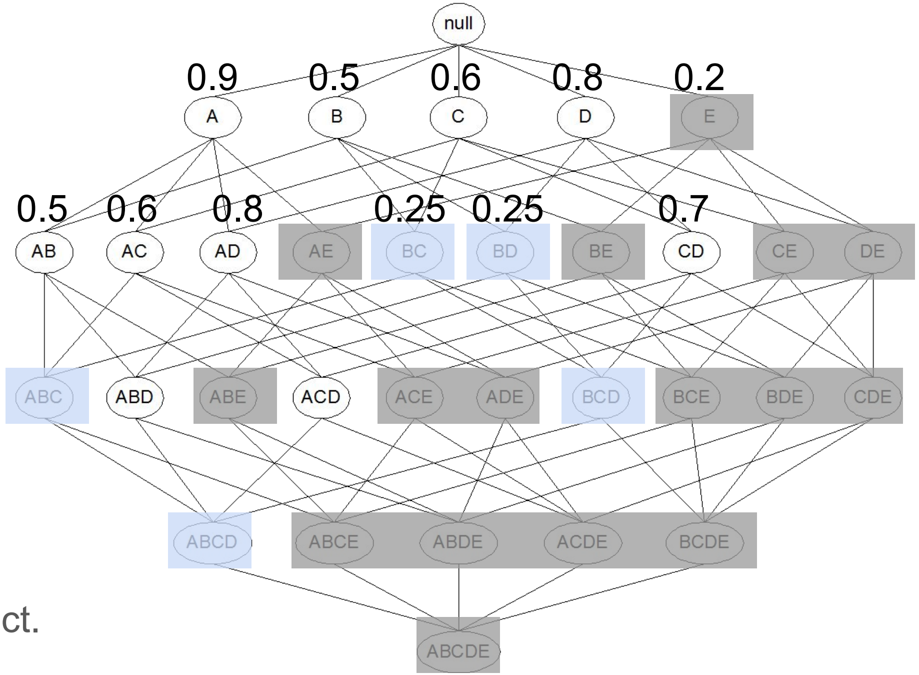
- Advantage
  - Algorithm is simple and comprehensible.

- Disadvantage
  - Runtime increases exponentially with the amount of item. => *FP-Growth*
  - It shows only correlations, not cause and effect.

3. Apriori

- What is Apriori?
  - To reduce the number of item set in association analysis.
  - The case, which includes infrequent item, will also not be frequent.

- How does it work?
  1. Make one-item-set frequent set with items.
  2. Select cases above the minimum support.
  3. Make two-item-set with survived items.
  4. Select cases above the minimum support
  5. Repeat until k-item-set



- Advantage
  - Algorithm is simple and comprehensible.

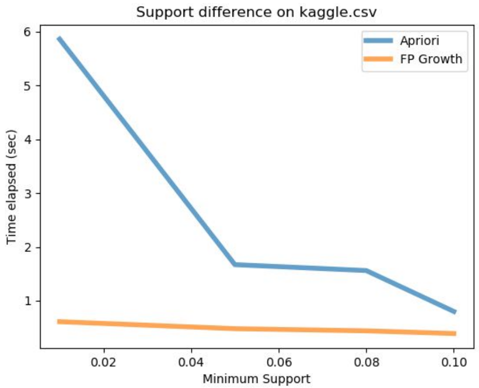
- Disadvantage
  - Runtime increases exponentially with the amount of item. => *FP-Growth*
  - It shows only correlations, not cause and effect.

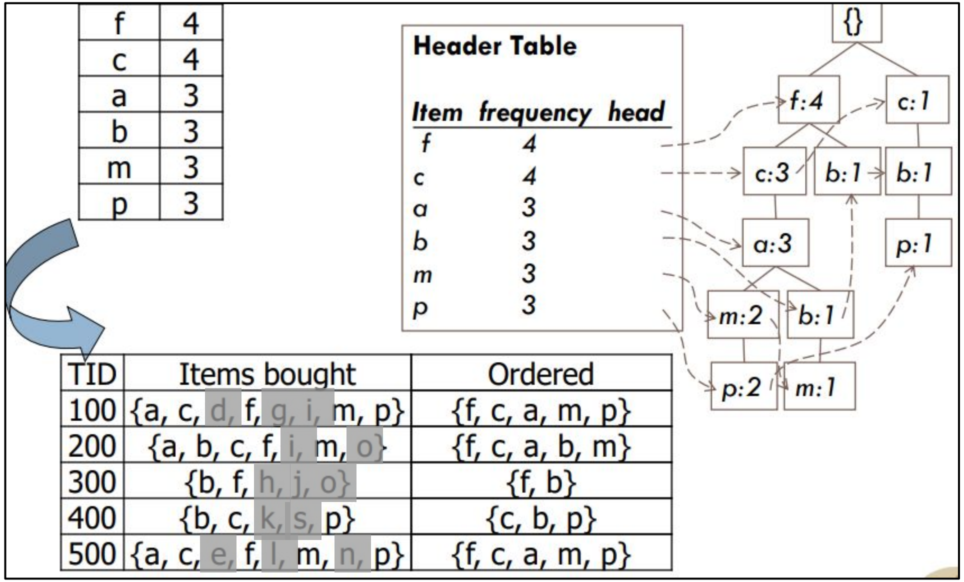
4. FP-Growth

- What is FP-Growth?
  - Better speed(computation) with similar performance compare to Apriori
  - FP(Frequent Pattern) Tree structure

- Advantage
  - Faster than Apriori
  - No need to make Item-set

- Disadvantage
  - Algorithms are difficult to design.





f	4
c	4
a	3
b	3
m	3
p	3

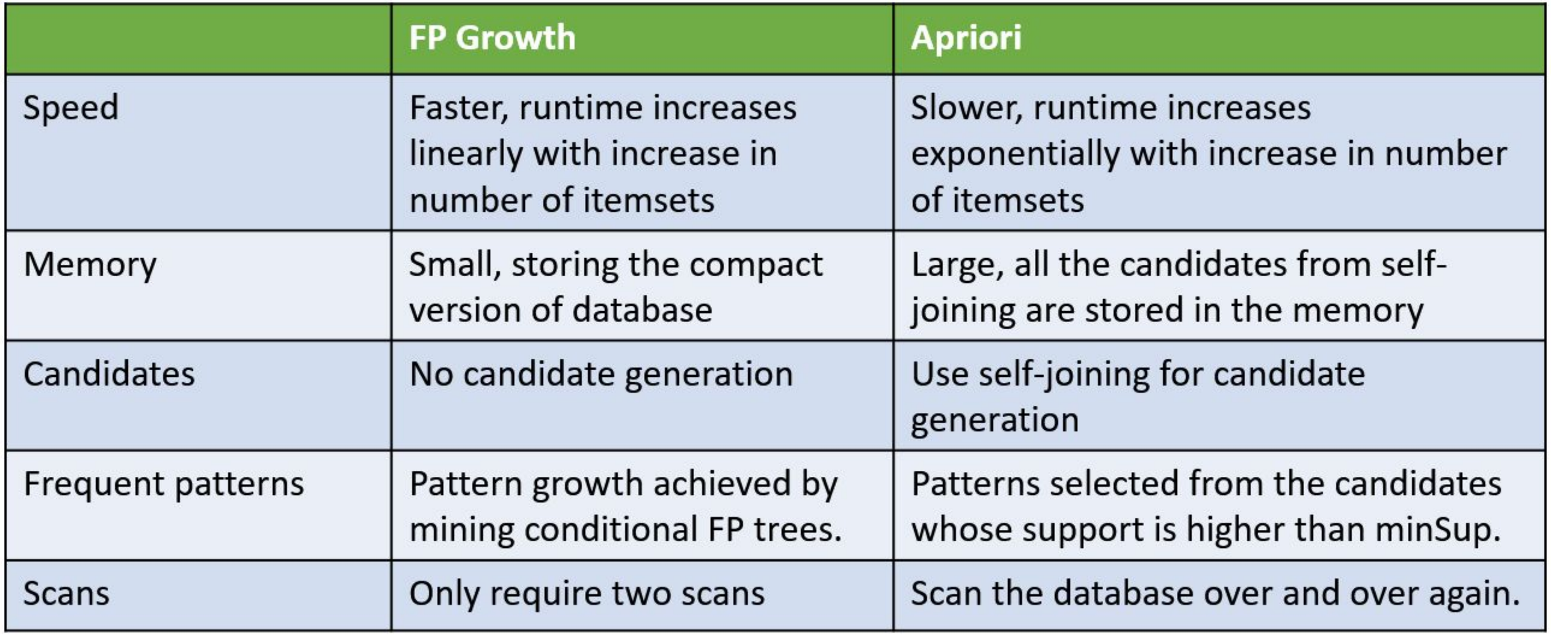
Header Table

Item	*frequency*	*head*
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	

TID	Items bought	Ordered
100	{a, c, d, f, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, i, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, c, e, f, l, m, n, p}	{f, c, a, m, p}

5. Apriori vs FP-Growth

- Comparison



	FP Growth	Apriori
Speed	Faster, runtime increases linearly with increase in number of itemsets	Slower, runtime increases exponentially with increase in number of itemsets
Memory	Small, storing the compact version of database	Large, all the candidates from self-joining are stored in the memory
Candidates	No candidate generation	Use self-joining for candidate generation
Frequent patterns	Pattern growth achieved by mining conditional FP trees.	Patterns selected from the candidates whose support is higher than minSup.
Scans	Only require two scans	Scan the database over and over again.

https://towardsdatascience.com/fp-growth-frequent-pattern-generation-in-data-mining-with-python-implementation-244e561ab1c3