МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**
Кафедра **ПЗ**

РЕФЕРАТ

На тему: *“Дослідження роботи алгоритму apriori для напівструктурованих даних”*

З дисципліни: *“ Методи та засоби досліджень в ІПЗ”*

Лектор:
професор каф. ПЗ
Федасюк Д.В.

Виконав:
ст. гр. ПЗІП-12
Гаврилюк А.М.

Прийняв:
ст. викл. каф. ПЗ
Марусенкова Т.А.

« ____ » _____ 2017 р.

∑= ____ .

Львів 2017

ЗМІСТ

ВСТУП

ІТ є однією з найбільш динамічніших сфер, активно розробляються нові технології, створюються сотні програмних продуктів та накопичаються приблизно декілька сотень петабайт інформації у день.

Big Data - набори інформації (структурованої, напіструктурованої та неструктурованої) настільки великих розмірів, що традиційні способи та підходи (здебільшого засновані на рішеннях класу бізнесової аналітики та системах управління базами даних) не можуть бути застосовані до них. Альтернативне визначення називає великими даними феноменальне прискорення нагромадження даних та їх ускладнення. Важливо також відзначити те, що часто під цим поняттям у різних контекстах можуть мати на увазі як дані великого об'єму, так і набір інструментів та методів (наприклад, засоби масово-паралельної обробки даних системами категорії NoSQL, алгоритмами MapReduce, чи програмними каркасами проекту Hadoop). Багато інструментів, призначених для аналізу великих даних, можуть обробляти неструктуровані дані.

Кінцевою метою цієї обробки є отримання результатів, які легко сприймаються людиною та є ефективними в умовах безперервного росту й розподілення інформації по численних вузлах обчислювальної мережі.

Для характеристики великих даних використовують «три v»: їх обсяг (англ. volume), швидкість накопичення нових даних та їх обробки (англ. velocity) та різноманіття типів даних, які можуть оброблятися (англ. variety).

До основних переваг використання технології можна віднести:

- отримання якісно нових знань за рахунок комплексного аналізу усієї інформації у єдиному аналітичному сховищі;
- розширення функціональності існуючих інформаційних систем підтримки бізнесу;
- збільшення ефективності використання апаратних ресурсів серверів;
- забезпечення мінімальної вартості використання всіх видів інформації за рахунок можливості використання ПЗ з відкритим кодом і хмарних технологій.

Критика великих даних пов'язана з тим, що їх зберігання не завжди приводить до отримання вигоди, а швидкість оновлення даних і «актуальний» часовий інтервал не завжди розумно порівнянні.

Переважно будь-яка сучасна система функціонує власне на базі певних предметних та інформаційних технологій. Впровадження і використання інтелектуальних систем бізнес-аналітики в компаніях дозволяє їм набувати конкурентоспроможності за рахунок накопичення інтелектуальної інформації та покращення ефективності процесу прийняття рішень. Опрацювання даним алгоритмом структурованих та напівструктурованих даних дозволяє оцінити його роботу, показати на яких засобах та на яких типах даних його використовувати ефективніше та доцільніше. А також розроблювана система дозволить працівникам сфер торгівлі чи послуг приймати оптимальні рішення щодо свого бізнесу. Прогнозувати обсяги виготовлення тої чи іншої продукції, або ж надання певних послуг в залежності від їх попиту серед клієнтів. Допоможе знайти певні закономірності для подальшого опрацювання бізнес-аналітиками.

Напівструктуровані дані – це форма структурованих даних, яка не відповідає формальній структурі моделей даних, пов'язаних з реляційними базами даних або іншими формами таблиць даних, але, тим не менш, містить мітки або інші маркери для відокремлення семантичних елементів та дотримання ієрархії записи та поля в даних.

У напівструктурованих даних об'єкти, що належать до одного класу, можуть мати різні атрибути, навіть якщо вони згруповані разом, а порядок атрибутів не є важливим.

Напівструктуровані дані все частіше трапляються з моменту появи Інтернету, де повнотекстові документи та бази даних більше не є єдиними формами даних, а для різних додатків потрібен носій для обміну інформацією. У об'єктно-орієнтованих базах даних часто зустрічаються напівструктуровані дані. Прикладами таких даних є XML та JSON.

АЛГОРИТМ APRIORI

Apriori — алгоритм глибинного аналізу даних щодо частих одиниць у множинах і машинного навчання щодо правил асоціювання, що застосовується переважно до баз даних транзакцій. Алгоритм ідентифікує елементи/одиниці, що часто повторюються у базі, і розширює їх список до все більших множин з дотриманням правила достатньої частотності. Визначені алгоритмом множини частих одиниць можна використати для визначення правил асоціювання, по яких стають помітними загальні тенденції в базі даних.

Це знаходить застосування в таких областях, як аналіз ринкового кошика. Одиницями при цьому є пропоновані товари, а покупка являє собою транзакцію, яка містить куплені предмети (одиниці). Алгоритм при цьому визначає кореляції такого виду:

Якщо хтось купує шампунь і лосьйон для гоління, у 90 % випадків купується також і піна для гоління.

Дані, що надаються до аналізу, складаються з таблиці транзацій (на рядках), в якій перераховуються будь-які бінарні одиниці (у колонках). Алгоритм Apriori знаходить співвідношення між множинами одиниць, які зустрічаються у великій частині транзакцій. Правила асоціювання, які отримуються в результаті, мають форму A –> B; при цьому A і B є множинами одиниць, а правило стверджує, що коли у великій частині транзакцій зустрічається множина одиниць A, то там часто зустрічається і множина одиниць B.

Асоціативні правила

Афінітивний аналіз (affinity analysis) — один з розповсюджених методів Data Mining. Його назва походить від англійського слова affinity, що у перекладі означає «близькість», «подібність». Ціль даного методу — дослідження взаємного зв'язку між подіями, які відбуваються спільно. Різновидом афінітивного аналізу є аналіз ринкового кошика (market basket analysis), ціль якого - виявити асоціації між різними подіями, тобто знайти правила для кількісного опису взаємного зв'язку між двома або більше подіями. Такі правила називаються асоціативними правилами (association rules).

Базовим поняттям у теорії асоціативних правил є ***транзакція*** — деяка множина подій, що відбуваються спільно. Типова транзакція - покупка клієнтом товару в супермаркеті. У переважній більшості випадків клієнт купує не один товар, а набір товарів, що називається ринковим кошиком. При цьому виникає питання: чи є покупка одного товару в кошику наслідком або причиною покупки іншого товару, тобто, чи пов'язані дані події? Цей зв'язок і встановлюють асоціативні правила. Наприклад, може бути виявлене асоціативне правило, котре стверджує, що клієнт, який купив молоко, з імовірністю 75 % купить і хліб.

Наступне важливе поняття — ***предметний набір***. Це непорожня множина предметів, що з'явилися в одній транзакції.

Аналіз ринкового кошика - це аналіз наборів даних для певної комбінації товарів, пов'язаних між собою. Іншими словами, виконується пошук товарів, присутність яких у транзакції впливає на ймовірність наявності інших товарів або комбінацій товарів.

Сучасні касові апарати в супермаркетах дозволяють збирати інформацію про покупки, що може зберігатися в базі даних. Потім накопичені дані можуть використовуватися для побудови систем пошуку асоціативних правил.

У табл. 1.1 представлений простий приклад, що містить дані про ринковий кошик. У кожному рядку вказується комбінація продуктів, придбаних за одну покупку. Хоча на практиці доводиться мати справу з мільйонами транзакцій, у яких беруть участь десятки й сотні різних продуктів, приклад обмежений 10 транзакціями, що містять 13 видів продуктів: аби проілюструвати методику виявлення асоціативних правил, цього досить.

Таблиця 1. Приклад набору транзакцій

№	Транзакція
1	Сливи, салат, помідори
2	Селера, цукерки
3	Цукерки
4	Яблука, морква, помідори, картопля, цукерки
5	Яблука, апельсини, салат, цукерки, помідори
6	Персики, апельсини, селера, помідори
7	Квасоля, салат, помідори
8	Апельсини, салат, морква, помідори, цукерки
9	Яблука, банани, сливи, морква, помідори, цибуля, цукерки
10	Яблука, картопля

Візуальний аналіз прикладу показує, що всі чотири транзакції, у яких фігурує салат, також включають помідори, і що чотири із семи покупок, що

містять помідори, також містять салат. Салат і помідори в більшості випадків купуються разом. Асоціативні правила дозволяють виявляти й кількісно описувати такі збіги.

Асоціативне правило складається із двох наборів предметів, що мають назву умова (antecedent) та наслідок (consequent), й записуються у вигляді $X \to Y$. Таким чином, асоціативне правило формулюється у вигляді: «Якщо умова, то наслідок».

Умова може обмежуватися тільки одним предметом. Правила звичайно відображаються за допомогою стрілок, спрямованих від умови до наслідку, наприклад, помідори → салат.

Візуальний аналіз прикладу показує, що всі чотири транзакції, у яких фігурує цибулю, також включають картоплю, і що чотири із семи шести, що містять картоплю, також містять цибулю. Цибулю і картоплю в більшості випадків купуються разом. Асоціативні правила дозволяють виявляти й кількісно описувати такі збіги. Також правила можуть бути комбіновані {Цибуля, Картопля} => {Бургер}

Асоціативне правило складається із двох наборів предметів, що мають назву умова (antecedent) та наслідок (consequent), й записуються у вигляді $X \to Y$. Таким чином, асоціативне правило формулюється у вигляді: «Якщо умова, то наслідок».

Умова може обмежуватися тільки одним предметом. Правила звичайно відображаються за допомогою стрілок, спрямованих від умови до наслідку, наприклад, цибуля → картопля.

Основними характеристиками, що описують асоціативне правило, є підтримка (support) і вірогідність (confidence).

Якщо позначити базу даних транзакцій через D, а число транзакцій у цій базі N, то кожна транзакція $d_i, \quad i = 1 \ldots N$ являє собою певний набір предметів. Позначимо підтримку правила через S, а вірогідність – через C.

Підтримка асоціативного правила — це число транзакцій, які містять як умову, так і наслідок. Наприклад, для асоціації $A \to B$ можна записати

$$S(A \to B) = P(A \cap B) = \frac{n(\{A; B\} \in d_i)}{N} \qquad (1.1)$$

Вірогідність асоціативного правила $A \to B$ являє собою міру точності правила й визначається як відношення кількості транзакцій, що містять умову і наслідок, до кількості транзакцій, що містять тільки умову:

$$C(A \to B) = P(A \mid B) = \frac{n1(\{A; B\} \in d_i)}{n1(\{A\} \in d_i)} \qquad (1.2)$$

Якщо підтримка й вірогідність досить високі, можна з великою ймовірністю стверджувати, що будь-яка майбутня транзакції, що включає умову, буде містити й наслідок.

Для прикладу розглянемо правило салат → помідори, що напрошується з попередніх спостережень . Для нього

$S(салат \to помідори) = 4/10 = 0{,}4$;

$C(салат \to помідори) = 4/4 = 1$.

Дане правило, що зустрічається в 40% транзакцій, є для вихідних даних абсолютно вірним – у всіх випадках, коли клієнт купує салат, він разом з тим купує й помідори. Це легко пояснити логічно - обидва продукти використовуються для готування овочевих блюд і дійсно часто купуються разом.

Тепер розглянемо асоціацію $конфеты \to помидоры$, у якій містяться слабко сумісні в гастрономічному плані продукти.

Підтримка даної асоціації S = 4/10 = 0,4 – та ж, що в попереднього правила, а вірогідність C = 4/6 = 0,67. Таким чином, порівняно невисока вірогідність даної асоціації дає привід засумніватися в тому, що вона є правилом.

Число 0,67 здається не таким вже й малим. Чому ж ми говоримо про «незначну вірогідність» цієї асоціації? Справа в тому, що помідори зустрічаються в 7 чеках з 10 ($P(B) = 0{,}7$). Прийнято, що всі правила з вірогідністю меншою, ніж проста ймовірність випадання наслідку не повинні

розглядатися, адже вони, по суті, випадкові. Для прийняття асоціативного правила його вірогідність повинна бути не меншою ніж імовірність наслідку.

Останнє зауваження при досить великій номенклатурі товарів призводить до необхідності подвоювати кількість розрахунків. На практиці аналітики можуть віддавати перевагу правилам, які мають високу підтримку (вище певного рівня, наприклад, 0,3) або високу вірогідність (не менше 0,8-0,85). Висування одночасних вимог щодо підтримки й вірогідності дозволяють значно позм'якшити критерії (підтримку до 0,1-0,15, вірогідність – до 0,67-0,75). Правила, для яких значення підтримки й вірогідності перевищують певні, задані користувачем пороги, називають сильними правилами (strong rules). Всі наведені вище числові значення - емпіричні. Наприклад, у задачах виявлення шахрайських операцій значення підтримки може знижуватися й до 1%, оскільки із шахрайством пов'язане порівняно невелике число транзакцій.

Крім об'єктивних оцінок (підтримки й вірогідності) кожного зі згенерованих правил, різні джерела радять використовувати деякі суб'єктивні оцінки. Всі вони, так чи інакше, базуються на об'єктивних.

Ліфт (від interest lift – підвищення інтересу) обчислюється в такий спосіб

$$L(A \to B) = C(A \to B) / P(B) \qquad (1.3)$$

Ліфт - це відношення частоти появи умови в транзакціях, які містять й умову, і наслідок, до частоти появи наслідку в цілому. Чим більше значення ліфта, тим частіше наслідок визначається умовою в порівнянні з випадками, коли умова відсутня. Якщо ліфт дорівнює 1, зв'язок відсутній, близькі ж до нуля значення свідчать про сильну зворотну залежність.

Для нашого прикладу в таблиці 1.1. візьмемо два правила з однаковою вірогідністю:

$P(салат) =$ 4/10 = 0,4. $C(помидоры \to салат) =$ 4/7 = 0,57.

$P(конфеты) =$ 6/10 = 0,6. $C(помидоры \to конфеты) =$ 4/7 = 0,57.

Здавалося б, правила однаково достовірні. Після розрахунку ліфта все стає на свої місця:

$L(помидоры \to салат)$ = 0,57/0,4 = 1,425;

$L(\text{помидоры} \rightarrow \text{конфеты})$ = 0,57/0,6 = 0,95.

Не варто вважати ліфт універсальним мірилом адекватності. Справа в тому, що правило з меншою підтримкою й більшим ліфтом може бути менш значимим, ніж альтернативне правило з більшою підтримкою й меншим ліфтом. Це пов'язане з тим, що останнє застосовується для більшого числа покупців.

Левередж (leverage – важіль, плече) – це різниця між спостережуваною частотою, з якої умова й наслідок з'являються спільно, та добутком частот появи умови й наслідку окремо

$$T(A \rightarrow B) = S(A \rightarrow B) - P(A) \cdot P(B) \qquad (1.4)$$

Левередж дозволяє впоратися із ситуаціями, коли й підтримка, й ліфт у правил ідентичні, але їх значимість явно відрізняється. Наприклад, у нашому овочевому магазині в правила $\text{салат} \rightarrow \text{помідори}$

$C = 1;\ S$ = 0,7; L = 1/0,7 = 1,43.

Морква, як показує таблиця 1.1, також продається тільки з помідорами, і також зустрічається чотири рази, тому й у правила $\text{морква} \rightarrow \text{помідори}$

$C = 1;\ S$ = 0,7; L = 1/0,7 = 1,43.

А от левередж у цих правил відрізняється на 30%:

$T(\text{морква} \rightarrow \text{помідори}) =$ 0,3 – 0,3 * 0,7 = 0,09.

$T(\text{салат} \rightarrow \text{помідори}) =$ 0,4 – 0,4 * 0,7 = 0,12.

Таким чином, значимість другої асоціації більша, ніж першої.

Альтернативою левередж є поліпшення.

Поліпшення (improvement) – це відношення частоти спостережуваних виконань правила до добутку частот появи умови й наслідку окремо.

$$I(A \rightarrow B) = \frac{S(A \rightarrow B)}{P(A) \cdot P(B)} \qquad (1.5)$$

фактично, поліпшення показує, у скільки разів розглянуте правило забезпечує правильний прогноз краще, ніж випадкове вгадування. Всі правила $I(A \to B) \leq 1$ не є значимими.

Такі міри, як ліфт, левередж і поліпшення, можуть використовуватися для обмеження набору розглянутих асоціацій шляхом встановлення граничних значень значимості, нижче яких асоціації відкидаються.

Принцип роботи алгоритму

Таблиця 2. Приклад набору транзакцій

Transaction ID	Цибуля	Картопля	Бургер	Молоко	Пиво
t_1	1	1	1	0	0
t_2	0	1	1	1	0
t_3	0	0	0	1	1
t_4	1	1	0	1	0
t_5	1	1	1	0	1
t_6	1	1	1	1	1

Ключовим поняттям в алгоритмі Апріорі є антимонотонність міри підтримки. Це передбачає що:

- Всі підмножини частої Набір повинні бути частими
- Подібним чином, для будь-якого незвичного набору, всі його суперсеті також повинні бути нечастими

Давайте розглянемо інтуїтивно зрозуміле пояснення алгоритму за допомогою наведеного вище прикладу(табл. 2). Перед початком процесу, встановіть порогової підтримки до 50%, тобто лише ті елементи значні, для яких підтримка більше 50%.

Крок 1: Створити таблицю частот для всіх елементів, що виникають у всіх транзакціях. Для нашого випадку:

Одиниця	**Частота**
Цибуля(O)	4
Картопля(P)	5
Бургер(B)	4
Молоко(M)	4
Пиво(Be)	2

Крок 2: Ми знаємо, що важливі лише ті елементи, для яких підтримка більше або дорівнює пороговій підтримці. Тут границя підтримки становить 50%, отже, лише ті елементи значущі, що трапляються у більш ніж трьох транзакціях, а такими є - Цибуля (O), Картопля (P), Бургер (B) і Молоко (M). Тому ми отримуємо наступну таблицю:

Одиниця	Частота
Цибуля(O)	4
Картопля(P)	5
Бургер(B)	4
Молоко(M)	4

Крок 3: Наступним кроком є зробити всі можливі пари значущих елементів, маючи на увазі, що порядок не має значення, тобто AB такий самий, як BA. Щоб зробити це, візьміть перший предмет і з'єднайте його з усіма іншими, такими як OP, OB, OM. Аналогічним чином розглянемо другий елемент і об'єднає його з попередніми пунктами, тобто PB, PM. Ми розглядаємо лише попередні пункти, оскільки PO (як OP) вже існує. Отже, всі пари в нашому прикладі - OP, OB, OM, PB, PM, BM.

Крок 4: Тепер ми розглянемо випадки кожної пари у всіх транзакціях.

Набір	Частота
OP	4
OB	3
OM	2
PB	4
PM	3
BM	2

Крок 5. Знову тільки ті набори наборів є значними, які перетинають порогову підтримку, а такі - OP, OB, PB та PM.

Крок 6. Тепер скажімо, ми хотіли б шукати набір з трьох елементів, які купуються разом. Ми будемо використовувати пункти, знайдені на кроці 5, і створимо набір з 3 елементів.

Щоб створити набір з 3 елементів, потрібно інше правило, яке називається самоз'єднання. Він говорить, що з пар OP, OB, PB та PM ми шукаємо дві пари з однаковою першою буквою, і об’єднуємо їх:

- OP та OB, це дає OPB
- PB і PM, це дає PBM

Далі ми знайдемо частоту для цих двох наборів.

Набір	Частота
OPB	4
PBM	3

Знову застосувавши правило порогу, ми виявили, що OPB є єдиним значним набором елементів.

Отже, набір з 3 предметів, який був придбаний найчастіше, - OPB.

Приклад, який ми розглянули, був досить простим, і видобуток частих предметів у товаристві зупинено на 3 пунктах, але на практиці існує кілька десятків предметів, і цей процес може тривати багато предметів. Припустимо, що ми отримали значні набори з трьома елементами як OPQ, OPR, OQR, OQS та PQR, і тепер ми хочемо створити набір з 4 елементів. Для цього ми розглянемо набори, які мають перші два алфавіти, тобто загалом

- OPQ та OPR дають OPQR
- OQR та OQS дають OQRS

Взагалі, ми повинні шукати набори, які тільки відрізняються в їхній останній букві / предметі.

Тепер, коли ми розглянули приклад функціональності алгоритму Апріорі, давайте сформулювати загальний процес.

Загальний процес алгоритму Apriori

Весь алгоритм можна розділити на два етапи:

Крок 1: застосуйте мінімальну підтримку, щоб знайти всі часто використовувані набори з елементами k у базі даних.

Крок 2. Використовуйте правило самопідключення, щоб знайти часті множини з елементами k + 1 за допомогою часто використовуваних k-елементів. Повторіть цей процес з k = 1 до моменту, коли нам не вдається застосувати правило самопідключення.

Такий підхід до розповсюдження частих елементів, що належить до одного, називається підходом "знизу вверх".

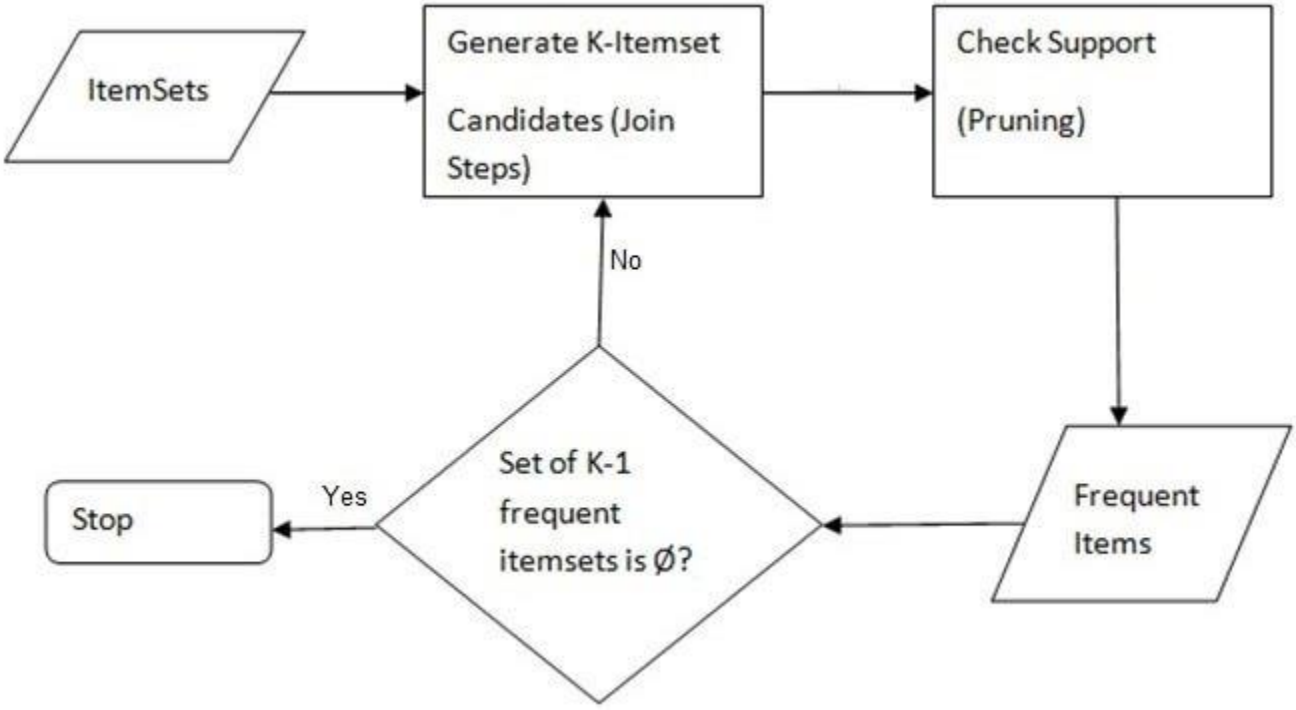

Рис. 1. Приклад роботи алгоритму у вигляді блок схеми

Видобування асоціативних правил

До цих пір ми розглянули алгоритм Апріорі по відношенню до частого покоління пакетів. Існує ще одне завдання, за допомогою якого ми можемо використовувати цей алгоритм, тобто ефективно знаходити правила об'єднання.

Щоб знайти правила об'єднання, нам потрібно знайти всі правила, що мають підтримку, більшу за порогову підтримку та довіру, більшу за порогову довіру.

Але як ми знаходимо це? Одним із можливих шляхів є груба сила, тобто перелік всіх можливих правил асоціації та обчислення підтримки та впевненості для кожного правила. Потім скасуйте правила, які не підтримують порогову підтримку та довіру. Але це обчислювально дуже важкий і заборонений, оскільки кількість всіх можливих правил асоціації зростає експоненціально з кількістю предметів.

Враховуючи, що в наборі I є n пунктів, загальна кількість можливих правил асоціації становить 3 ^ n - 2 ^ (n +1) + 1.

Ми також можемо використовувати інший спосіб, який називається двоетапним підходом, для пошуку ефективних правил асоціації.

Двоступеневий підхід:

Крок 1: генерація частотних наборів елементів: знайдіть всі набори предметів, для яких підтримка перевищує порогову підтримку відповідно до процесу, який ми вже бачили раніше в цій статті.

Крок 2: генерація правил: створюйте правила з кожної частинки, використовуючи бінарний розділ часто використовуваних предметів і шукайте тих, що мають високу впевненість. Ці правила називаються правилами кандидатів.

Давайте розглянемо наш попередній приклад, щоб отримати ефективне правило об'єднання. Ми виявили, що OPB є частим комплектом. Тому для цієї проблеми, крок 1 вже зроблено. Отже, давайте розглянемо крок 2. Усі можливі правила, що використовують OPB, є:

OP⟶B, OB⟶P, PB⟶O, B⟶ OP, P⟶OB, O⟶PB

Якщо X являє собою частий набір елементів з k-елементами, то існують правила асоціації кандидатів 2 ^ k-2.

Плюси алгоритму:

1. Легкий у реалізації та розумінні.
2. Може бути використаний для величезних даних.

Мінуси алгоритму:

1. Іноді, потрібно знайти величезну кільуість правил кандидатів, що є обчислювано затратно.
2. Підрахуном підтримки також затратний по ресурсах, адже потрібно пройтись по всій базі.

HADOOP

Apache Hadoop — вільна програмна платформа і каркас для організації розподіленого зберігання і обробки наборів великих даних з використанням моделі програмування MapReduce, при якій завдання ділиться на багато дрібніших відособлених фрагментів, кожен з яких може бути запущений на окремому вузлі кластера, що складається з серійних комп'ютерів. Всі модулі в Hadoop спроектовані з врахуванням припущення, що злам апаратного забезпечення трапляється часто і це повинно автоматично враховуватись фреймворком.

Ядро системи Apache Hadoop складається з розподіленої файлової системи Hadoop Distributed Filesystem (HDFS), та системи обчислень на основі моделі програмування MapReduce. Hadoop розділяє файли на великі блоки і розподіляє їх між вузлами кластера. Тоді він передає запакований код на вузли для паралельної обробки даних. Цей підхід користується локальністю даних, коли вузли маніпулюють лише даними до яких мають доступ. Це дозволяє обробляти набір даних швидше і ефективніше ніж в традиційнішій суперкомп'ютерній архітектурі яка покладається на паралельну файлову систему в якій обчислення та дані для них передаються через високошвидкісну мережу.

Основний фреймворк Apache Hadoop складається з наступних модулів:

Hadoop Common — містить бібліотеки та утиліти потрібні іншим модулям Hadoop;

Hadoop Distributed File System (HDFS) — розподілена файлова система, яка зберігає дані на звичайних машинах, надаючи дуже високу загальну пропускну здатність на кластері загалом;

Hadoop YARN — платформа що відповідає за керування обчислювальними ресурсами в кластерах і їх використання для користувацьких завдань.

Hadoop MapReduce — реалізація моделі програмування MapReduce для обробки великих об'ємів даних.

З часом, термін Hadoop почав вживатись не тільки щодо вищезгаданих базових модулів та підмодулів, а й до «екосистеми», тобто набору додаткових пакетів програмного забезпечення, які можуть встановлюватись поверх, або поряд з Hadoop, наприклад таких як Apache Pig[en], Apache Hive, Apache HBase, Apache Phoenix, Apache Spark, Apache ZooKeeper, Cloudera Impala, Apache Flume, Apache Sqoop, Apache Oozie, та Apache Storm.

MapReduce та HDFS в Apache Hadoop's були натхненними статтями Google про їх алгоритм MapReduce та Google File System.

Фреймворк Hadoop написаний переважно на Java, з частиною системного коду на C та утилітами командного рядка як shell скрипти. Хоча в програмах MapReduce звичайним є код на Java, для реалізації «map» та «reduce» частин користувацької програми можна використовувати будь-яку мову програмування завдяки «Hadoop Streaming». Інші проекти в екосистемі Hadoop надають багатші інтерфейси користувача.

Hadoop активно використовується у великих промислових проектах, надаючи можливості, аналогічні платформі Google Bigtable/GFS/MapReduce, при цьому компанія Google офіційно делегувала Hadoop та іншим проектам Apache право використання технологій, на які поширюються патенти, пов'язані з методом MapReduce. Одним з найбільших користувачів і розробників Hadoop є компанія Yahoo!, вона активно використовує цю систему в своїх пошукових кластерах (Hadoop-кластеру Yahoo, що складається з 40 тисяч вузлів, належить світовий рекорд швидкості сортування великого обсягу даних). Hadoop-кластер

використовується в Facebook для обробки однієї з найбільших баз даних, в якій зберігається близько 30 петабайт інформації. Hadoop також лежить в основі платформи Oracle Big Data і активно адаптується компанією Microsoft для роботи з СУБД SQL Server, Windows Server і хмарній платформі Azure Cloud з метою створення нових продуктів для організації розподіленої обробки великих обсягів даних. Hadoop є одним з ключових ланок суперкомп'ютера IBM Watson, який виграв бій з найкращими гравцями телевізійної гри-вікторини "Jeopardy!".

ВИСНОВКИ

Після аналізу існуючих систем, що займаються пошуком закономірностей у даних, огляду існуючих підходів та алгоритмів для опрацювання великих структурованих і напівструктурованих даних було виявлено потребу у даному дослідженні. Опрацювання даним алгоритмом структурованих та напівструктурованих даних дозволяє оцінити його роботу, показати на яких засобах та на яких типах даних його використовувати ефективніше та доцільніше. А також розроблювана система дозволить працівникам сфер торгівлі чи послуг приймати оптимальні рішення щодо свого бізнесу. Прогнозувати обсяги виготовлення тої чи іншої продукції, або ж надання певних послуг в залежності від їх попиту серед клієнтів. Допоможе знайти певні закономірності для подальшого опрацювання бізнес-аналітиками.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Асоціативні правили. [Електронний ресурс] // Створення асоціативних правил : сайт. - Режим доступу http://victoria.lviv.ua/html/sss/associative.html
2. Almasi, G.S. and A. Gottlieb (2009). Highly Parallel Computing. Benjamin// Cummings publishers, Redwood City, CA. – 235c.
3. Harris, Dereck Intel jettisons its Hadoop distro and puts millions behind Cloudera (27 March 2014).
4. Уайт, Том Hadoop. Подробное руководство = Hadoop: The Definitive Guide. — 2-е. — СПб.: Питер, 2013. — 672 с.
5. Hadoop [Електронний ресурс] // Вікіпедія : сайт. - Режим доступу https://ru.wikipedia.org/wiki/Hadoop
6. MapReduce [Електронний ресурс] // Вікіпедія : сайт. - Режим доступу http://en.wikipedia.org/wiki/MapReduce#Map_function
7. Mapreduce Appliance. [Електронний ресурс] // MapReduce : сайт. - Режим доступу http://www.teradata.com/products/Aster_MapReduce_Appliance
8. Big Data Applience. [Електронний ресурс] // Oracle Big Data: сайт. - Режим доступу https://www.oracle.com/engineered-systems/big-data-appliance/index.html
9. GreenPlum. [Електронний ресурс] //: сайт. - Режим доступу http://www.emc.com/campaign/global/greenplumdca/index.htm
10. Big Data. [Електронний ресурс] // Вікіпедія: сайт. - Режим доступу https://en.wikipedia.org/wiki/Big_data
11. Дюк В., Самойленко А. Data Mining: Учебный курс. – СПб: Питер, 2001. – 368 с.
12. А. Шахиди. Data Mining – добыча данных [Електронний ресурс] // Big Data: сайт. - Режим доступу // http://www.basegroup.ru

13. Karen Montgomery. Big Data Now: 2014 Edition. O'Reilly Media.- Junuary, 2015 – 165p.
14. Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data. John Wiley & Sons. 2014-12-19. 300p.
15. Apache Hadoop. [Електронний ресурс] // Big Data: сайт. - Режим доступу https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
16. Tong Qiang, Zhou Yuanchun, Wu Kaichao, Yan Baoping. A quantitative association rules mining algorithm[J]. Computer engineering. 2007
17. Zhu Yixia, Yao Liwen, Huang Shuiyuan, Huang Longjun. A association rules mining algorithm based on matrix and trees[J]. Computer science. 2006, 33(7):196-198
18. Hadoop File System. [Електронний ресурс] // hadoop-distributed-file-system: сайт. - Режим доступу https://ru.wikipedia.org/wiki/Hadoop
19. Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules.In Proc.20th Int. Conf. Very Large Data Bases, 487-499.
20. Hadoop File System. [Електронний ресурс] // hadoop-distributed-file-system: сайт. - Режим доступу https://www.safaribooksonline.com/blog/2013/02/13/the-hadoop-distributed-file-system/