УДК 004.378

Т.А. ГРИГОРОВА, В.П. ЛЯШЕНКО, О.О. МОСКАЛЕНКО
Кременчуцький національний університет імені Михайла Остроградського

# ДОСЛІДЖЕННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ПОШУКУ ІНФОРМАЦІЇ

*Перевагами використання машинного навчання в пошуку є те, що пошукова система може навчатися і тим самим приводити до більш персоналізованих відповідей, а не поширених результатів. У відомих пошукових системах такі алгоритми використовуються вже давно і постійно удосконалюються. У роботі на прикладах було досліджено методи і алгоритми машинного навчання, які використовуються для пошуку інформації, їх переваги і недоліки. Було обрано колоборативну фільтрацію, кластерізацію та пошук асоціативних правил. Розглянуті основні підходи колоборативної фільтрації – кореляційні і латентні моделі. В якості кореляційних моделей – фільтрацію за подібністю користувачів (user-based filtration) і фільтрація за подібністю посилань (item-based filtration). Ці моделі розглянуті на прикладах, які показують, як працюють алгоритми. Фільтрація за подібністю посилань прогнозує оцінку на основі оцінок іншого посилання та використовує регресійний аналіз або, як альтернативу, використовує спрощений предиктор, що називається алгоритмом SlopeOne. Розглянуті метрики: евклідова відстань, косинусний коефіцієнт та коефіцієнт кореляції Пірсона, що використовуються для визначення коефіцієнта подібності користувачів в моделі фільтрації за подібністю користувачів. В якості латентних моделей розглянуті алгоритми кластеризації: бікластеризація, алгоритм просторової кластеризації з присутністю шуму DBSCAN, алгоритм нечіткої кластеризації c-means. Всі ці алгоритми призначені для формування кластерів даних за визначеним критерієм. Розглянуто пошук асоціативних правил на прикладі алгоритму Apriori, що генеруються на основі всіх поширених пошукових наборів, виявлених в базі даних пошукових запитів, які задовольняють заданому критерію відповідності. Для застосування цього алгоритму дані були приведені до бінарного вигляду та відповідної структури даних. Зроблено висновки, що кожний з цих методів має свої недоліки і тільки завдяки їх комбінуванню можна досягти бажаного результату для підвищення якості пошуку в залежності від задач, які поставив замовник.*

*Ключові слова: машинне навчання, кластеризація, колаборативна фільтрація, пошук асоціативних правил.*

Т.А. ГРИГОРОВА, В.П. ЛЯШЕНКО, А. А. МОСКАЛЕНКО
Кременчугский национальный университет имени Михаила Остроградского

# ИССЛЕДОВАНИЕ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПОИСКА ИНФОРМАЦИИ

*Преимуществами использования машинного обучения для поиска является то, что поисковая система может учиться и тем самым приводить к более персонализированным ответам, а не распространенным результатам. В известных поисковых системах такие алгоритмы используются уже давно и постоянно совершенствуются. В работе на примерах были исследованы методы и алгоритмы машинного обучения, которые используются для поиска информации, их преимущества и недостатки. Были выбраны колоборативная фильтрация, кластеризация и поиск ассоциативных правил. Рассмотрены основные подходы колоборативной фильтрации – корреляционные и латентные модели. В качестве корреляционных моделей – фильтрацию по сходству пользователей (user-based filtration) и фильтрация по сходству ссылок (item-based filtration). Эти модели рассмотрены на примерах, которые показывают, каким образом работают алгоритмы. Фильтрация по сходству ссылок прогнозирует оценку на основе оценок другой ссылки и использует регрессионный анализ или, в качестве альтернативы, использует упрощенный предиктор, который называется алгоритмом SlopeOne. Рассмотрены метрики: евклидово расстояние, косинусный коэффициент и коэффициент корреляции Пирсона, которые используются для определения коэффициента сходства пользователей в модели фильтрации по сходству пользователей. В качестве латентных моделей рассмотрены такие алгоритмы кластеризации, как бикластеризация, алгоритм пространственной кластеризации с присутствием шума DBSCAN, алгоритм нечеткой кластеризации c-means. Все эти алгоритмы предназначены для формирования кластеров данных по определенному критерию. Рассмотрен поиск ассоциативных правил на примере алгоритма Apriori, генерируемых на основе всех часто задаваемых поисковых наборов, выявленных в базе данных поисковых запросов, которые удовлетворяют заданному*

*критерию соответствия. Для применения этого алгоритма данные были приведены к бинарному виду и соответствующей структуре данных. Сделаны выводы, что каждый из этих методов имеет свои недостатки и только благодаря их комбинированию можно достичь желаемого результата для повышения качества поиска в зависимости от задач, которые поставил заказчик.*

*Ключевые слова: машинное обучение, кластеризация, колаборативная фильтрация, поиск ассоциативных правил.*

T.A. HRYHOROVA, V.P. LYASHENKO, O.O. MOSKALENKO
Kremenchug National University named after Mykhailo Ostrogradsky

## RESEARCH OF MACHINE LEARNING METHODS FOR SEARCH INFORMATION

*The advantages of using machine learning in search are that the search engine can learn and thus lead to more personalized answers, rather than the common results. In well-known search engines, such algorithms have been used for a long time and are constantly being improved. In the work on the examples were studied methods and algorithms of machine learning, which are used to search for information, their advantages and disadvantages. Collaborative filtering, clustering, and search for associative rules were chosen. The main approaches of collaborative filtering - correlation and latent models are considered. The correlation models - user similarity filtering (user-based filtration) and link similarity filtering (item-based filtration). These models are considered in the examples, which show how the algorithms work. Link similarity filtering predicts an estimate based on the estimates of another link, and uses regression analysis or, alternatively, uses a simplified predictor called the SlopeOne algorithm. The metrics Euclidean distance, cosine coefficient and Pearson correlation coefficient, which are used to determine the user similarity coefficient in the filtering model by user similarity, are considered. Clustering algorithms such as biclasterization, DBSCAN noise clustering algorithm, and fuzzy c-means fuzzy clustering algorithm are considered as latent models. All these algorithms are designed to form data clusters according to a certain criterion. The search for associative rules is considered on the example of the Apriori algorithm, which is generated on the basis of all frequent search sets found in the database of search queries that meet the specified match criterion. To apply this algorithm, the data were reduced to a binary form and the corresponding data structure. It is concluded that each of these methods has its drawbacks and only by combining them can achieve the desired result to improve the quality of the search depending on the tasks set by the customer.*

*Keywords: machine learning, clustering, collaborative filtering, search for associative rules.*

### Постановка проблеми

З кожним днем в пошукових системах індексується все більше інформації, а тому і зростає кількість інформаційного сміття. Для пошуку інформації в мережі Інтернет актуальним стає використання алгоритмів штучного інтелекту для визначення більш точного результату та відсіювання зайвої інформації для пошукового запиту. Дослідження методів інтелектуального пошуку необхідно для підвищення ефективності пошуку.

### Аналіз останніх досліджень і публікацій

До найбільш відомих методів машинного навчання, які використовуються для пошуку, належать колаборативна фільтрація, кластеризації та пошук асоціативних правил [1]. Застосування підходів колаборативної фільтрації в пошукових системах використовується для відображення тематичних результатів пошуку на основі інтересів користувача, що формуються з історії пошуку, або груп інтересів других користувачів, що виконують схожий пошуковий запит [2]. Кластеризація розділяє множину даних на підмножини за схожістю деяких характеристик [2]. Дані всередині одного кластера повинні бути схожими на інші дані того ж самого кластеру, та відрізнятись від даних інших кластерів. Кластеризація є найбільш поширеною формою машинного навчання без вчителя[3]. Пошук асоціативних правил дозволяє знаходити закономірність між зв’язаними об’єктами. Асоціативні правила були створені для сегментації покупців по поведінці під час покупок, але цим їх застосування не обмежується і вони

використовуються в інформаційних пошукових системах для видачі більш релевантних результатів пошуку.

**Мета дослідження**

Метою дослідження є аналіз методів машинного навчання – колаборативної фільтрації, кластеризації та пошуку асоціативних правил для подальшого їх використання для пошуку інформації та підвищення якості пошуку в залежності від запитів користувача.

**Основна частина**

Колаборативна фільтрація має два основних підходи [3]: кореляційні моделі (Memory-Based Collaborative Filtering) та латентні моделі (Latent Models for Collaborative Filtering).

Для кореляційних моделей використовується сімейство алгоритмів Slope One для аналізу різноманітних побажань користувачів і вироблення персональних рекомендацій. Кореляційний підхід має дві основні моделі: фільтрація за подібністю користувачів (user-based filtration) – базується на основі інтересів та оцінок інших користувачів системи, що мають схожу групу інтересів, та фільтрація за подібністю посилань (item-based filtration) – базується на порівнянні результатів пошукової видачі різних користувачів. Алгоритми колаборативної фільтрації працюють наступним чином: переглядають велику множину людей і відшукують в ній підмножину зі схожими інтересами, об’єднують переваги та створюють ранжований список пропозицій [4].

Якщо застосовувати кореляційний підхід для пошукової системи, то в якості інтересів будемо формувати таблицю з посилань користувача.

Розглянемо приклад реалізації кореляційного підходу. Перше, що необхідно – це спосіб представлення користувачів та їх переваг. Якщо застосовувати кореляційний підхід для пошукової системи, то в якості інтересів будемо формувати таблицю з посилань користувача. Наприклад, рейтинг посилань можна оцінювати в залежності від того, відкрив користувач посилання чи ні. Якщо користувач відкрив посилання, ставимо оцінку 1, якщо не відкрив, ставимо оцінку 0.

Створюємо таблицю в базі даних. В таблиці повинна міститись інформація про користувача, який відкрив посилання (id користувача), інформація про посилання, яке відкрив користувач (id посилання), поле для пошукового запиту та поле для рейтингу посилання (0 або 1).

Тобто, в інформаційній системі є користувачі, про яких відома інформація про те, які посилання відкрив користувач для пошукового запиту .

Табл. 1

Приклад таблиці для зберігання інформації про користувачів

| Користувач (id) | Посилання (id) | Рейтинг (0 або 1) | Пошуковий запит(id) |
|---|---|---|---|
| Користувач 1 | 1,3,5,7 | 1 | 1 |
| Користувач 1 | 2,4,6 | 0 | 1 |
| Користувач 2 | 1,2,4,5,7 | 1 | 2 |
| Користувач 2 | 3,6 | 0 | 2 |
| … | … | … | … |
| Користувач n | 4,5 | 1 | 50 |

Принцип полягає в тому, що, якщо користувач виконує пошуковий запит, який виконував інший користувач та про якого є вже дані в таблиці, то можна припустити, що користувачу можуть підійти деякі результати пошукового запиту.

Колаборативна фільтрація за подібністю посилань прогнозує оцінку на основі оцінок іншого посилання, та використовує регресійний аналіз $f(x) = ax + b$.

Тобто, якщо ми маємо 100 посилань, то може бути до 100000 лінійних регресій для вивчення до 200000 регресорів. Такий підхід може бути неефективним через перенавчання, тому необхідно вибирати посилання, для яких відома оцінка багатьох користувачів. Альтернативою може бути використання спрощеного предиктору (наприклад $f(x) = x + b$), що називається алгоритмом SlopeOne, в якому предиктор – це середня різниця між оцінками обох предметів. Щоб застосувати алгоритм SlopeOne для заданих n-предметів, необхідно розрахувати та зберегти середню різницю та кількість голосів для кожної з $n^2$ пар предметів.

Для реалізації такого алгоритму (Slope One) можна використати бібліотеку OneSlopeOne [5]. Для цього створимо таблицю «oso_user_ratings», яка буде містити id користувача, id предмета (посилання) та рейтинг посилання (0 або 1), та заповнимо таблицю.

Табл. 2

Таблиця з рейтингом посилань користувачів «oso_user_ratings»

| user_id | item_id | rating |
|---|---|---|
| 1 | 1 | 1.0000 |
| 1 | 2 | 1.0000 |
| 2 | 1 | 1.0000 |
| 2 | 2 | 1.0000 |
| 3 | 1 | 1.0000 |

З таблиці 2 ми бачимо, що користувач 1 та користувач 2 переглядали посилання 1 та 2. Користувач 3 переглядав посилання 1. Якщо виконати алгоритм SlopeOne для користувача 3, то система запропонує користувачу посилання 2, так як з його посиланням 1 користувач 1 та користувач 2 переглядали також посилання 2.

Для знаходження пропозицій для користувача, в алгоритмі SlopeOne використовується наступний SQL-запит:

*$sql = 'select s.item_id2 from oso_slope_one s,oso_user_ratings u where u.user_id = '. $userId . ' and s.item_id1 = u.item_id and s.item_id2 != u.item_id group by s.item_id2 order by sum(u.rating * s.times – s.rating)/sum(s.times) desc limit ' . $limit;*

Такий алгоритм має значний недолік: зі зростанням кількості користувачів також зростає складність обчислення рекомендацій.

Прикладом алгоритму колаборативної фільтрації за посиланнями є запатентований алгоритм «item_to_item» компанії Amazon, що розраховує подібність посилань як косинус між векторами переглядів в матриці користувачів та посилань [6].

$$similarity(\vec{A}, \vec{B}) = \cos(\vec{A}, \vec{B}) = \frac{\vec{A} \times \vec{B}}{\left|\left|\vec{A}\right|\right| * \left|\left|\vec{B}\right|\right|}.$$

В даному алгоритмі використовується один коефіцієнт на кожну пару посилань (косинус), на основі якого і створюються рекомендації [6].

Розглянемо його роботу на прикладі статистики переглядів посилань користувачами зображених у таблиці 3.

Табл. 3

Статистика переглядів посилань користувачами

| Користувач | Посилання 1 | Посилання 2 | Посилання 3 |
|---|---|---|---|
| Користувач 1 | Переглянув | Не переглядав | Переглянув |
| Користувач 2 | Не переглядав | Переглянув | Переглянув |
| Користувач 3 | Не переглядав | Переглянув | Не переглядав |

В даній таблиці, якщо користувач переглянув посилання, записуємо 1, якщо не переглянув, запи суємо 0. Косинус між «Посиланням 1» та «Посиланням 2» розраховується за формулою [6]:

$$\cos(1,2) = \frac{(1,0,0) \times (0,1,1)}{||(1,0,0)|| * ||(0,1,1)||} = 0 .$$

Косинус між «Посиланням 1» та «Посиланням 3» розраховується так:

$$\cos(1,3) = \frac{(1,0,0) \times (1,1,0)}{||(1,0,0)|| * ||(1,1,0)||} = \frac{1}{\sqrt{2}} \approx 0.71 .$$

Косинус між «Посиланням 2» та «Посиланням 3» розраховується так:

$$\cos(2,3) = \frac{(0,1,1) \times (1,1,0)}{||(0,1,1)|| * ||(1,1,0)||} = \frac{1}{2} \approx 0.5 .$$

Таким чином, користувач, який знаходиться на сторінці «Посилання 1», отримає в рекомендацію «Посилання 3», на сторінці «Посилання 2» – «Посилання 3», на сторінці «Посилання 3» – «Посилання 1». В даному алгоритмі використовується один коефіцієнт на кожну пару посилань (косинус), на основі якого і створюються рекомендації [6].

Недоліками алгоритму «item_to_item» є: тривіальні рекомендації – пропонуються тільки популярні посилання, не враховуються інтереси конкретного користувача, проблема «холодного старту» – система рекомендує тільки ті посилання, які відвідували користувачі, та не рекомендує нові посилання.

Щоб враховувати інтереси конкретного користувача та видавати рекомендації на основі переваг інших користувачів, необхідно визначити, наскільки інтереси користувачів схожі. Для цього користувач порівнюється з іншими користувачами та вираховується коефіцієнт подібності. Для підрахування коефіцієнта подібності існує ряд метрик: евклідова відстань, косинусний коефіцієнт, коефіцієнт кореляції Спірмана, коефіцієнт Танімото, логарифмічна правдоподібність та інші [4].

Розглянемо кілька метрик коефіцієнта подібності. Одна із самих простих метрик – це евклідова відстань. В даному випадку посилання, які оцінили користувачі, або посилання, на які користувачі переходили, зображуються в вигляді координатної осі. На координатній осі розташовуються точки, що відповідають користувачам. Потім порівнюють дані точки, наскільки близько одна точка знаходиться від інших, ті користувачі, що знаходяться ближче від інших і є користувачами з подібними інтересами. Тобто, чим ближче два користувача на координатній осі, тим більш схожі в них інтереси. Щоб підрахувати відстань між одним користувачем та другим, необхідно скористатись формулою.

$$f(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}\,.$$

Ця метрика повертає результат від 0 до 1. Чим більше коефіцієнт, тим більш схожі інтереси у користувачів.

Косинусний коефіцієнт являє собою косинус кута між двома векторами, утвореними точкою $O(0,0,\dots)$ та точками користувачів $X(x_1,x_2,\dots)$, $Y(y_1,y_2,\dots)$. Необхідно виміряти подібність двох векторів $X$ та $Y$ [4]. Для цього потрібно створити обмеження косинусу кута $-1 \le \cos A \le 1$. Якщо $\cos A = 1$, то користувачі мають подібність по інтересам, якщо $\cos A = -1$, то користувачі мають протилежні інтереси. Косинусний коефіцієнт вираховується за формулою [4]:

$$K = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2}}.$$

Коефіцієнт кореляції Пірсона показує наскільки добре два набори даних лягають на пряму. Даний коефіцієнт ефективний коли дані погано нормалізовані. Коефіцієнт кореляції Пірсона набуває значень $K\,(-1 \le K \le 1)$, що показує тенденцію двох множин чисел, що йдуть попарно. Якщо залежність лінійна, то коефіцієнт кореляції $K = 1$, якщо зворотна, то $K = -1$, якщо залежності немає, то $K = 0$. Коефіцієнт кореляції Пірсона вираховується за формулою [4]:

$$K = \frac{N\sum_{i=1}^{N} x_i y_i - (\sum_{i=1}^{N} x_i)(\sum_{i=1}^{N} y_i)}{\sqrt{(N\sum_{i=1}^{N} x_i^2 - (\sum_{i=1}^{N} x_i)^2)(N\sum_{i=1}^{N} y_i^2 - (\sum_{i=1}^{N} y_i)^2)}}.$$

Розглянемо алгоритм колаборативної фільтрації user-based більш детально. Нехай у нас є матриця переваг користувача. Для порівняння користувачів будемо використовувати один з методів подібності. Тоді алгоритм колаборативної фільтрації полягає в наступному [6]:

1. Вибрати $n$ користувачів, переваги (інтереси) яких найбільш схожі на переваги користувача, який сформував запит. Для цього, для кожного з користувачів необхідно обчислити міру подібності одним із методів, та вибрати тих користувачів, коефіцієнт подібності в яких буде найбільшим.

Тобто, будемо мати дані у вигляді таблиці:

Табл. 4

Таблиця коефіцієнтів подібності користувачів

| | Користувач 1 | Користувач 2 | Користувач n | Сума |
|---|---|---|---|---|
| Користувач, що розглядається | K1 | K2 | Kn | K1+K2+Kn |

2. Для кожного користувача треба помножити його оцінки на обчислену величну міри. Таким чином, оцінки більш схожих користувачів будуть сильніше впливати на підсумкову позицію посилання.

3. Для кожної переваги порахувати суму каліброваних оцінок $n$ найбільш близьких користувачів, отриману суму розділити на суму $n$ обраних користувачів.

В вигляді формули такий алгоритм можна представити так [5,7]:

$$r_{u,i} = k \sum_{u' \in U} sim(u, u') r_{u',i} .$$

В даній формулі: $sim$ – обраний коефіцієнт подібності двох користувачів, $u$ – множина користувачів, $r$ – оцінка, $k$ – нормуючий коефіцієнт. Нормуючий коефіцієнт можна записати у вигляді формули:

$$k = 1 / \sum_{u' \in U} \left| sim(u, u') \right| .$$

Даний алгоритм має значний недолік – алгоритм підраховує рекомендації при кожному зверненні користувача.

До недоліків цих моделей можна віднести те, що необхідно постійно зберігати всю вихідну матрицю даних, а також є проблема холодного старту. Дані проблеми вирішують латентні моделі даних.

До латентних моделей належать алгоритми кластеризації.

Однією з латентних моделей є бікластеризація (ко-кластеризація) [8], методика аналізу даних, за допомогою якої можна одночасно кластеризувати як рядки, так і стовбці матриці. В задачах інформаційного пошуку бікластеризація застосовується для виявлення кластерів, що мають подібні властивості тільки за кількома ознаками [8]. Тобто, на вхід приймається матриця розміру $m \times n$, метод бікластеризації генерує бікластери – підмножину рядків, які виявляють подібну поведінку через підмножину стовбців.

Одним з найбільш використовуваних алгоритмів кластеризації є щільнісний алгоритм просторової кластеризації з присутністю шуму DBSCAN. На вхід даний алгоритм приймає матрицю близькості та два параметри: щільність точки – максимальна відстань між двома зразками, при якій можна вважати, що вони належать до однієї околиці; щільність околиці – кількість зразків в одній області, необхідних, щоб точка вважалась оточеною.

Якщо в базі даних є точки, які утворюють кластери різної щільності, то алгоритму не вдається добре кластеризувати точки даних, оскільки кластеризація залежить від параметру щільності околиці, який не може бути обраний окремо для всіх кластерів.

Якщо об'єкт належить до декількох кластерів в рівній мірі, існує алгоритм нечіткої кластеризації c-means, який визначає ймовірність того, що об'єкт належить до того чи іншого кластеру.

Для зберігання посилань користувачів можна використовувати асоціативні правила, які можна використовувати для видачі більш релевантних результатів пошуку. Тоді будемо зберігати інформацію у вигляді $U = \{u_1, u_2, u_3, \dots, u_n\}$ – множина користувачів, а $D$ – множина транзакцій (посилань) $T$, в якій кожна транзакція є набором з $U$, $T \subseteq U$. Кожна транзакція представляє собою бінарний вектор, де $t[k] = 1$, якщо $u_k$ присутнє в транзакції, інакше $t[k] = 0$. Асоціативним правилом називається імплікація $X \Rightarrow Y$, де $X \subset U, Y \subset U$ *та* $X \cap Y = \varnothing$ [9].

Для пошуку асоціативних правил будемо використовувати алгоритм Apriori, що генеруються на основі всіх частих предметних(пошукових) наборів, виявлених в базі даних транзакцій (пошукових запитів), які задовольняють заданому критерію відповідності. Для того, щоб застосувати даний алгоритм, необхідно привести

необхідні дані до бінарного вигляду та мати відповідну структуру даних. Дані про користувача та набір пошукових посилань можна представити у вигляді таблиці:

Табл. 5

Нормалізований вигляд асоціативних правил для алгоритму Apriori

| Транзакція | Посилання 1 | Посилання 2 | Посилання 3 | Посилання … |
|---|---|---|---|---|
| Користувач 1 | 0 | 1 | 1 | … |
| Користувач 2 | 0 | 0 | 0 | … |
| Користувач 3 | 1 | 1 | 1 | … |
| Користувач 4 | 1 | 0 | 0 | … |
| Користувач … | … | … | … | … |

В якості транзакцій можна зберігати дані пошукових запитів користувачів, а замість стовбців посилань будуть стовбці пошукових запитів.

Кількість стовбців в таблиці дорівнює множині транзакцій. Всі елементи таблиці повинні бути впорядковані в алфавітному порядку, або в числовому порядку зростання. Алгоритм Apriori працює в два етапи: на першому кроці необхідно знайти набір елементів, які часто зустрічаються, на другому кроці виявити в них правила. Для виявлення наборів елементів, які часто зустрічаються, алгоритм Apriori використовує властивість підтримки, в якому підтримка будь-якого набору елементів не може перевищувати мінімальної підтримки будь-якого елементу з його підмножини. Дана властивість алгоритму слугує для зниження розмірності простору пошуку. Тобто, з ростом розміру набору елементів, підтримка залишається такою ж або навіть зменшується. Набір із $k$ елементів буде зустрічатись часто тоді, коли всі його $k-1$ елементів підмножини будуть зустрічатись часто.

На першому кроці алгоритм Apriori підраховує одно-елементні набори, які часто зустрічаються. Для цього необхідно пройти по всьому набору даних і підрахувати для них підтримку, тобто, скільки разів набір зустрічається в базі даних. Далі алгоритм генерує набори елементів, які потенційно можуть часто зустрічатись (кандидати) та підраховує підтримку для потенційних наборів. Кожен кандидат $C_k$ формується шляхом розширеного набора із $k-1$ елементів, додаванням елементів із другого $k-1$-елементного набору. Крок об'єднання можна представити в вигляді SQL-запиту [9]:

*INSERT INTO* $C_k$ *SELECT* $p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}$ *FROM* $F_{k-1}p, F_{k-1}q$ *WHERE* $p.item_1 = q.item_1, p.item_2 = q.item_2, \dots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$.

Після генерації кандидатів необхідно підраховувати підтримку для кожного кандидата. Самим тривіальним способом є порівняння кожної транзакції з кожним кандидатом. Ефективнішим та швидшим є підхід, коли кандидати зберігаються в хеш-дереві. Внутрішні вузли дерева містять хеш-таблиці з покажчиками на нащадків, а листя – на кандидатів.

Хеш-дерево будується кожен раз, коли формуються кандидати. Кожен раз, коли формується новий кандидат, він заноситься в корінь дерева, і так далі, поки кількість кандидатів не досягне деякої межі. Коли кількість кандидатів стає більше межі, корінь перетворюється в хеш-таблицю, тобто стає внутрішнім вузлом, а для нього формуються потомки-листя. Щоб підрахувати підтримку для кожного кандидата, необхідно застосувати на кореневому рівні хеш-функцію $C_k \cap T_i = C_k$. Потім, на -рівні хеш-функція застосовується для елементів $k$-рівня. І так до тих пір, поки не досягнемо листка. Якщо кандидат, що зберігається в листку, є підмножиною розглянутої транзакції, тоді збільшуємо лічильник підтримки для цього кандидата на одиницю.

Після того, як знайдені всі набори елементів, які часто зустрічаються, алгоритм переходить до генерації правил. Щоб витягти правило з набору $F$, який часто зустрічається, необхідно знайти всі його непусті підмножини. Для кожної підмножини $s$ формуємо правило $s \Rightarrow (F - s)$ [9].

Візьмемо дані деяких користувачів, що виконували пошукові запити. В масив даних помістимо набір id пошукових запитів користувачів з одного кластеру:

*$users = ['User1' => ['12', '13', '15','22','25'], 'User2' =>['23', '24', '26'], 'User3' =>['11', '12', '15'], 'User4' =>['15', '12', '13'], 'User5' =>['23', '24', '26','27']];*

За допомогою методу Apriori спрогнозуємо пошуковий запит для користувача, який виконує, наприклад пошуковий запит з id 23. Скористаємось бібліотекою «php/ml», та виконаємо наступний програмний код:

```
$samples = ['User1' => ['12', '13', '15','22','25'], 'User2' =>['23', '24', '26'], 'User3' =>['11', '12', '15'], 'User4' =>['15', '12', '13'], 'User5' =>['23', '24', '26','27']];
$associator = new Apriori($support = 0.3, $confidence = 0.3);
$labels = [];
 $associator->train($samples, $labels);
$res = $associator->predict(['23']);
```

В даному коді змінна *$support* – це мінімальний рівень підтримки, *$confidence* – це мінімальний рівень достовірності. За допомогою функції *predict* прогнозуємо ймовірний асоціативний набір для пошукового запиту з id 23. Результат виконання коду показано на рис. 1.

**Для пошукового запиту з id 23 можливі наступні результати запитів:**

**id 24**

**id 26**

array(2) { [0]=> string(2) "24" [1]=> string(2) "26" }

**Рис. 1. Результат застосування алгоритму Apriori для прогнозування**

Якщо користувач не здійснював пошукових запитів, але перейшов на сторінку пошуку, то можна пропонувати посилання користувачів з його кластеру, які здійснювали пошукові запити, та на основі цих пошукових запитів виводити набір який зустрічається досить часто. Тобто, таким чином можна вирішувати проблему «холодного старту». Алгоритм Apriori в нашому випадку буде пропонувати id 12 та id 15 (рис. 2).

array(4) { ["antecedent"]=> array(1) { [0]=> string(2) "12" } ["consequent"]=> array(1) { [0]=> string(2) "15" } ["support"]=> float(0.6) ["confidence"]=> float(1) }

**Id з пошукового набору, які зустрічаються частіше інших:**

**12**

**15**

**Рис. 2. Результат застосування алгоритму Apriori для прогнозування**

## Висновки

У даній роботі було проаналізовано методи машинного навчання, які використовують для пошуку. Були розглянуті алгоритми колоборативної фільтрації, та пошуку асоціативних правил на прикладах. Було з'ясовано, що всі вони мають свої недоліки. Щоб зробити роботу цих алгоритмів більш ефективною, тобто звести до мінімуму негативні наслідки, їх потрібно комбінувати в залежності від задачі, яку потрібно розв'язувати.

## Список використаної літератури

1. Щербаков Д. Как искусственный интеллект повлиял на поисковые системы. URL: https://www.uplab.ru/blog/artificial-intelligence/

2. Segaran T., *Programming Collective Intelligence* (O'Reilly Media Inc., California, 2007), pp. 27–46.

3. Yao Z., Weibin C., "Review of research on collaborative filtering recommendation", *Micro Machines and Applications 6*, 2013, pp. 4-10.

4. Owen S., Anil R., Dunning T. and Friedman E., *Mahout in Action* (Manning Publications Co, Shelter Island, 2012), pp. 48–56.

5. Pu Wang and HongWu Ye, "A Personalized Recommendation Algorithm Combining Slope One Scheme and User Based Collaborative Filtering", *IIS* '09, 2009, pp. 152-154.

6. Bo F. and Jiujun C. "Collaborative filtering and recommendation algorithm based on multiple similarities among users", *Computer Science*, No.39, 2012, pp. 23-26.

7. Hofmann T. and Puzicha J., "Latent class models for collaborative filtering", in *Proceedings of the International Joint Conference on Artificial Intelli*gence, 1999, pp. 668–693.

8. Madeira S. C. and Oliveira A. L., "Biclustering Algorithms for Biological Data Analysis: A Survey", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, VOL 1, NO. 1, pp. 24-45 January-March 2004.

9. Bhavithra, J. and Saradha, A. Personalized Web Page Recommendation Using Case-Based Clustering and Weighted Association Rule Mining. *Cluster Computing*, 2019, 22, 6991-7002

## References

1. Sherbakov, D. Kak iskusstvenny intellekt povliyal na poiskovye sistemy. Retrieved from https://www.uplab.ru/blog/artificial-intelligence/

2. Segaran T., *Programming Collective Intelligence* (O'Reilly Media Inc., California, 2007), pp. 27–46.

3. Yao Z., Weibin C., "Review of research on collaborative filtering recommendation", *Micro Machines and Applications 6*, 2013, pp. 4-10.

4. Owen S., Anil R., Dunning T. and Friedman E., *Mahout in Action* (Manning Publications Co, Shelter Island, 2012), pp. 48–56.

5. Pu Wang and HongWu Ye, "A Personalized Recommendation Algorithm Combining Slope One Scheme and User Based Collaborative Filtering", *IIS* '09, 2009, pp. 152-154.

6. Bo F. and Jiujun C. "Collaborative filtering and recommendation algorithm based on multiple similarities among users", *Computer Science*, No.39, 2012, pp. 23-26.

7. Hofmann T. and Puzicha J., "Latent class models for collaborative filtering", in *Proceedings of the International Joint Conference on Artificial Intelli*gence, 1999, pp. 668–693.

8. Madeira S. C. and Oliveira A. L., "Biclustering Algorithms for Biological Data Analysis: A Survey", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, VOL 1, NO. 1, pp. 24-45 January-March 2004.

9. Bhavithra, J. and Saradha, A. Personalized Web Page Recommendation Using Case-Based Clustering and Weighted Association Rule Mining. *Cluster Computing*, 2019, 22, 6991-7002

Григорова Тетяна Альбертівна − к.т.н., доцент, доцент кафедри інформатики і вищої математики Кременчуцького національного університету імені Михайла Остроградського, e-mail: grital0403@gmail.com, ORCID: 0000-0002-4371-8624.

Ляшенко Віктор Павлович – д.т.н., професор, завідувач кафедри інформатики і вищої математики Кременчуцького національного університету імені Михайла Остроградського, e-mail: viklyash2903@gmail.com, ORCID:0000-0002-4538-631X

Москаленко Олександр Олександрович − аспірант кафедри інформатики і вищої математики Кременчуцького національного університету імені Михайла Остроградського, e-mail: alexsashamosk@gmail.com.