Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра математики

Магістерська робота

освітній ступінь – магістр

на тему: «**СИСТЕМИ ПІДТРИМКИ РІШЕНЬ ПРИ КЛАСИФІКАЦІЇ ОБ'ЄКТІВ**»

Виконав: студент 2-го року навчання
освітньо-наукової програми «Системний аналіз», спеціальності 124 Системний аналіз
Коваленко Руслан Вікторович

Керівник: Чорней Р. К.,
кандидат фіз.-мат. наук, доцент

Рецензент _____
(прізвище та ініціали)

Магістерська робота захищена
з оцінкою_____
СекретарЕК_____
«____» _____2021р.

Київ – 2021

Зміст

Анотація

на кваліфікаційну/магістерську роботу на тему: «Систематизація та класифікація на основі метричного підходу»

Магістрант: Коваленко Руслан Вікторович.

Керівник: кандидат фіз.-мат. наук, доцент Чорней Руслан Костянтинович.

Робота виконана на 36 сторінках, містить вступ, 2 розділи, висновки та 9 використаних джерел.

Мета роботи: систематизація та кластеризація продуктів із різних інтернет-платформ на основі неконтрольованої відповідності (unsupervised matching) та метричного способу.

В розділі 1 проведено аналіз та підготовку даних, досліджено та сформовано семантичні правила для визначення міри важливості токенів.

В розділі 2 спроектовано числову оціночну функцію, яка дозволяє на базі параметрів, таких як, частота використання токенів, довжини, відстані від початку назви товарів, а також “гарячих токенів”, визначених за правилами, описаними в першому розділі. Це дозволяє впорядкувати комбінації токенів в межах однієї назви товару, разом з тим, визначити комбінацію, яка найкраще представляє даних продукт. На базі цієї інформації, ми сформували первинні кластери.

В розділі 3 проведено верифікацію побудованих первинних кластерів в 2 розділі, враховуючи підтримувальні правила, сформованих задля відхилень від початкової задачі.

В результаті отримали алгоритм, який здатний кластеризувати товари незалежно від семантичної складності формулювання назви. А також, верифікувати, і разом з тим, реструктуризувати кластери у випадку неявних відхилень під час їх побудови.

Вступ

Актуальність. Дослідження продуктів за характеристикою, зокрема за назвою є досить цікавою що для користувачів, що для маркетів. Дуже часто, головною метою пошуку в інтернеті є порівняння цін та характеристик або отримання інформації, яка представлена іншими користувачами, наприклад, у вигляді відгуків. Однією із найголовніших причин агрегації великого обсягу інформації: конкурентної політики, цін та інших особливостей є організація рекламної кампанії. Існує великий попит на подібну структуризацію інформації, недавні дослідження демонструють спроби накопичувати важливу інформацію та збагачувати заголовки продуктів, використовуючи веб-пошукові системи. Такі методи передбачають здійснення запиту для кожного заголовку. Як тільки вся необхідна інформація буде зібрана, найважливіші слова, які зустрічалися в в результаті запитів, ідентифікуються та додаються до заголовків продуктів. Заключним є етап присвоєння числового значення до кожного слова за допомогою функції важливості, після цього, використовується додаткова функція схожості, яка дозволяє визначити, чи відносяться два або більше заголовків до однакового продукту. Як виявилося згодом, такий підхід не є ідеальним, оскільки має проблеми з масштабованістю, повільним виконанням запитів задля отримання всієї необхідної інформації, і найголовніше: недостача гнучкості [1], [2, с. 1].

Об’єкт дослідження: правила класифікації впорядкованого списку продуктів за їх характеристиками.

Предмет дослідження: класифікація товарів за принципом встановлення відповідностей без вчителя, що дозволяє виконати кластеризацію на холодних даних.

Головною метою цієї роботи є представлення кардинально іншої концепції кластеризації продуктів за їх назвою, яка позбавлена всіх

вищеописаних проблем. Ідея базується на морфологічному аналізі заголовків продуктів. Зокрема, роботу можна розділити на два етапи: в першому побудуємо всі комбінації слів заголовків кожного із продуктів, разом з тим, зберігаючи певні статистичні дані, такі як позиція, частота та інші; на другому етапі привласнюємо оцінку кожній комбінації. Комбінацію із найбільшою оцінкою позначаємо як кластер, який містить кожний продукт. Паралельно розглянути підхід кластеризації товарів, який базується на глибинному навчанні, результат роботи якого збігається із результатом нашого головного алгоритму. А також, розробити два незалежних додатки, які відтворюють наш головний алгоритм та підхід глибинного навчання для порівняння результатів та перевірки наступної гіпотези: підхід, який ми розглядаємо як основний, здатен кластеризувати товари в декілька разів ефективніше за всі інші, зокрема алгоритм глибинного навчання.

Методи дослідження: алгоритм побудований за принципом навчання без вчителя, такий підхід дозволяє запускати систему на холодних даних.

Практичне значення роботи. Система здатна класифікувати товари від різних магазинів, що дозволяє структурувати каталоги продуктів. Попит такої структуризації існує серед безлічі маркетплейсів та агрегаторів.

Розділ 1. Алгоритм безконтрольної кластеризації товарів (Unsupervised Product Matching)

Представимо Unsupervised Product Matching - алгоритм кластеризації продуктів, який складається із трьох етапів аналізу текстової частини продуктів.

Unsupervised Product Matching - концепт, який базується на принципі кластеризації продуктів за їх назвами. Ідея концепту полягає у побудові комбінацій зі слів - токенів, та присвоєнні числового значення кожній комбінації. Комбінація із найбільшим числовим значенням утворює ядро кластера, яке найкраще ідентифікує продукт. Продукти, які належать до одного і того самого кластеру представляють екземпляр однакового продукту.

На першому етапі, алгоритм займається морфологічним аналізом слів, вивчаючи особливості та характеристики, ідентифікує потенційно важливі токени: атрибути, моделі або нейтральні. Тобто, класифікує кожне поле текстової частини продуктів відносно форми та семантики. Таким віртуальним полям присвоюється значення, які використовуються в кінцевій функції для оцінки комбінації. Остаточна функція також враховуватиме позицію та частоту полів.

На етапі-верифікації, сформовані кластери аналізуються на blocking-умови, наприклад, одна із таких умов дозволяє запобігти двом сутностям, які належать до однакового магазину, міститися в одному кластері. Таким чином, якщо можливо сформувати blocking-умови, на практиці алгоритми кластеризації використовують дані умови на етапі верифікації. В розглянутій нами задачі такі умови дозволяють виключити дублікати продуктів, які містяться в каталозі магазину.

Варто зауважити, що підхід UPM не займається обрахуванням коефіцієнта схожості між двома продуктами, не визначає чи співпадають два продукти. Для того, щоб розібратися в алгоритмі UPM для кластеризації продуктів за їх назвами, проведемо ознайомлення із елементами алгоритму [3, с. 2-3].

1.1 Ознайомлення із елементами алгоритму

Нехай ми маємо список магазинів S_i, що включає різні харчові та нехарчові продукти. Кожен магазин розповсюджує певний каталог товарів із певною додатковою інформацією. Даний каталог використовується для ознайомлення споживачів з продукцією магазину. Інформація в каталозі є організованою в певні записи, які містить унікальний ідентифікатор, включаючи інформацію про бренд, об'єм та кількість.

Кожен представник, кожен магазин формує свій каталог незалежно один від одного. Саме тому, два продукти, які представляють один екземпляр, де один із магазину si може містити інформацію про бренд і певні атрибути в той самий час інший із магазину sj може не зберігати в собі такої інформації. Тобто, два представники можуть мати різний опис - назву для одного і того ж самого продукту. Тобто, задачею нашого unsupervised алгоритму подолати цю різницю в назвах продуктів.

Назва товарів можуть містить різного типу стрічки, специфікації та описи. Всякого роду інформацію із назв продуктів ми будемо називати токенами. Нехай W – список всіх токенів продукта t. Саме тому, k – комбінація C_k буде будь-яка підмножина W розмірністю k, без повторень і без урахування послідовності токенів.

Наприклад, нехай W_t складається із 3 токенів $-\{w_1, w_2, w_3\}$, відповідно можна сказати, що існує три можливих $2-k$ −комбінацій: $\{w_1, w_2\}$, $\{w_1, w_3\}$, $\{w_2, w_3\}$ та одну $3-k$ −комбінацію $-\{w_1, w_2, w_3\}$.

Отже, якщо назва t продукта складається із Lt токенів, тоді для того, що порахувати всі можливі k-комбінації, скористаємося наступною формулою:

$$N(l_t, k) = C_{l_t}^{k} = \frac{l_t!}{k!(l_t-k)!} \quad (1)$$

Складність такого обрахування для будь-якого k буде $O(2^{l_t})$.
Варто зауважити, що даний підхід відтворює концепцію N-грам. Зазвичай N −грам використовується в передбаченні на основі ймовірнісних моделей. n-грамою називають патерн або вікно розміром n, яке поступово ковзається зліва направо по аналізуючій стрічці. Таким чином, це вікно містить не завжди важливі слова. Мається на увазі, що важливі слова іноді не є сусідніми в стрічці. Хоч побудова всіх k −комбінацій є більш вартісною за n-граму, в той час вона є більш точною, беручи до уваги її здатність аналізувати не тільки сусідні токени.

- Оскільки обчислення всіх можливих комбінацій має експоненціальну складність, то головною задачею є обмеження побудови зайвих k-комбінацій наскільки це можливо. На щастя, розробка класифікатора показує, що продукти, які мають в середньому 6 - 11 токенів залежно від категорії та класу продуктів, і тільки певна частина цієї токенів дійсно несуть надважливу інформацію. Саме для цього ми обмежимо

обрахування k-комбінацій. Рахуватимемо тільки перші $2\ -, 3\ -, \ldots k\ -$ комбінації.

Відповідно, для назви товару t, який містить l_t токенів, загальну кількість комбінацій можна отримати скориставшись наступною формулою [3, с. 5] :

$$N_c(l_t,\ k) = \sum_{k=2}^{min\ (l_t,k)} C_{l_t}^{k} = \sum_{k=2}^{min\ (l_t,k)} \frac{l_t!}{k!(l_t-k)!} \qquad (2)$$

1.2 Аналіз назв продуктів

Одним із найважливіших і найкропіткіших процесів алгоритму є аналіз назв. Оскільки на даному етапі ми будуємо всі необхідні структури даних і збираємо необхідну інформацію для побудови кластерів. Саме тому даний етап ми розділимо на два підетапи, задачею першої частини очищення даних, визначення і видалення так званого шуму, який тільки ускладнюватиме роботу алгоритму та зменшуватиме ефективність і коректність кластеризації. Задачею наступного під етапу є проведення морфологічного аналізу назв: ідентифікація правил семантики токенів, а також, надати базову характеристику лексиконів - структур даних, які в майбутньому ми будемо використовувати для побудови кластерів.

1.2.1 Підготовка даних

Не завжди, кожна назва продуктів містить в собі однаково важливі токени або ідентифікатори, які допомагають нам відрізняти між собою

товари. Найчастіше, представники або магазини надають недоречну інформацію, потрібну для опису товарів. Такою інформацією є багатоканальні способи оплати, спеціальні знижки конкретних магазинів, додаткові пропозиції, способи доставки та інші можливості для споживача. Подібного роду інформацію ми буде класифікувати як надмірну та недоречну, з метою її оминанню під час аналізу токенів. Перед тим, як перейти до морфологічного аналізу текстової частини, проведемо також допоміжні дії з метою уникнення технічних розбіжностей між токенами. Наступні дії, які потрібно виконати для очищення нашого датасету з метою зведення до одного формату даних:

- Конвертація всіх символів назв товарів до нижнього регістру;
- Очищення від пунктуації. Варто зауважити, що певний список знаків є критичним для класифікації токенів, а саме “.”, “-” та “*”.
- Видалення дублікатів. Досить часто стрічки назв товарів містять дублікати токенів, видалення таких слів дозволить підвищити ефективність алгоритма [3, с. 5].

1.2.2 Морфологічний аналіз

Принцип Unsupervised Product Matching передбачає виділення найважливішої інформації, а саме ідентифікаторів, за допомогою яких будемо відрізняти продукти між собою - hot токени. А це буде черговим викликом в зв’язку із різними синтаксичними правилами, які використовували магазини незалежно один від одного. Задаче морфологічного аналізу є якраз визначення hot токенів. Для цього умовно класифікуємо всі можливі токени наступним чином:

- *Міксовані*, той випадок, коли токен містить і числа, і літери;
- *Числові*, даний вид токенів містить лише числові значення;
- *Алфавітний*, до даного виду належать всі токени, які ми не можемо віднести до першого або другого класу.

На наступному кроці ідентифікуємо важливі частини інформації серед токенів. Для цього визначимо певні категорії токенів:

- **Атрибути**. Ця група міститиме токени, за допомогою яких можливо відрізняти дуже схожі продукти. Наприклад, це стосується таких ситуацій, коли продукти відрізняються об'ємом:

  Продукт "Борошно А 0.5кг" відрізняється лише вагою від "Борошно В 1кг".

  Для цього ініціалізуємо локальний лексикон, відносно категорії продуктів. В останньому прикладі таким атрибутом є вага, виражена в кілограмах. Таким чином, токен буде належати до групи атрибутів тоді і тільки тоді, коли він сам або закінчення токену буде належати до завчасно визначеного лексикону("кг", "л", "шт", і т.д.). Досить часто спостерігається, що символ space між одиницями та значенням різнить атрибути. Проте, слід пам'ятати, що "0.5кг" та "0.5 кг" представляють однаковий атрибут.
- **Модель**. Наступна група токенів є найважливішою, оскільки саме вони представляють унікальність товарів. Досить часто каталоги різних магазинів містять моделі в різних формах, це ускладнює процес кластеризації, оскільки техніка unsupervised має вміти співставляти подібні речі, хоча форму вони можуть мати кардинально різну.

- **Нейтральна**. Токени, що не увійшли до першої або другої групи, будуть належати до третьої, нейтральної групи [3, с. 6].

Тип	Семантика	Ідентифікаційне правило
1	Атрибут	Числовий, після якого слідують одиниці виміру або міксований токен. В обидвох одиниці виміру належать до визначеного заздалегідь лексикону
2	Модель	Перший міксований токен, який не представляє Атрибут
3	Модель	Решта міксованих токенів, що зустрічаються в назві продукту
4	Модель	Числові токени, після яких не слідують одиниці виміру
5	Нейтральний	Решта токенів в назві, що не представляють групи Моделей та Атрибутів

Таблиця 1.2.2

1.2.3 Розробка структур даних

На даному етапі головною нашою задачею є збереження і структуризація тої важливої інформації, яку ми отримали під час морфологічного аналізу назв товарів. Оскільки процес алгоритму є

послідовним без паралельної обробки інформації, необхідно враховувати процеси структуризації та збереження даних. А також ефективне та практичне отримання інформації з бази даних.

Для цього введемо, для збереження всіх можливих токенів введемо структуру даних Token Lexicon. Елементом даного лексикону буде запис, котрий міститиме наступні дані:

- ідентифікатор токену;
- частоту, яка представлятиме кількість товарів, що включатимуть цей токен;
- значення токену;
- вид та тип семантики.

Для того, щоб зберігати комбінації, необхідно створити нові для цього умови - Combination Lexicon. Запис даної структури даних буде містити такі рядки:

- ідентифікатор комбінації;
- частота, відображатиме кількість продуктів, що міститимуть дану комбінацію;
- ключ-сигнатура, значення для ефективного пошуку комбінації;
- позиція. Ще одне ключове значення, необхідне подальшого обрахунку міри важливості комбінації. Це значення дорівнюватиме сумі відстаней токенів цієї комбінації від початку слова(Гіпотеза про порядок важливих слів в документі).

Значення сигнатури будемо обчислювати за допомогою спеціальної хеш-функції, яка на вхід приймає відсортований список ідентифікаторів

токенів, що належать даній комбінації. Таким чином значення хеш-функції будуть однаковим для однакового списку токенів. Саме це дозволить нам підвищити ефективність роботи з комбінаціями в структурі даних Combination Lexicon.

Останньою структурою даних, необхідною для повноцінної реалізації алгоритму буде Forward Index. Задачею цієї структури даних є поєднання продуктів із всієї інформацією, яку вдалося накопичити про товари. Тобто, Forward Index виконує роль лінкера між продуктом та записами в Token Lexicon і Combination Lexicon. Елементами такої структури будуть наступні значення:

- ідентифікатор товарів;
- указник на список відповідних комбінацій в Combination Lexicon;
- указник на список відповідних токенів в структурі Token Lexicon [3, с. 8-9].

Розділ 2. Принцип UPM

2.3 Етап 1. Відокремлення та збереження інформації

Методологія нашого алгоритму полягатиме в заповненні структур даних, які ми описали вище. Кожний товар p проходить поетапний аналіз назви. В першу чергу відбувається очищення стрічки від шуму - інформації, якої ми не потребуємо для аналізу та подальшої систематизації. На наступному кроці, стрічка із назвою товару розбивається на токени. Кожен токен проходить морфологічний аналіз, описаний в підрозділі 1.2.2. Крім того, механізм, який визначає тип та форму токенів, керується семантикою, зазначеною в Таблиця 1.2.2. Таким чином із зібраною інформацію про токен відбувається пошук посилання на токен в структуру Token Lexicon. У випадку, якщо структура не містить такого указника, відбувається збереження із значенням частоти рівним 1. В іншому випадку, відповідна частота в таблиці збільшується на 1. Після виконання даного етапу, ми маємо список указників на відповідні токени в Token Lexicon.

Продовжуючи алгоритм, відбувається побудова $2, 3, 4 \ldots n$ − комбінацій і генерування їх сигнатур. З метою покращення ефективності алгоритму, ми обмежуємо кількість комбінацій, які потрібні нам для кластеризації.

Після проведення всіх ітерацій алгоритму і детального аналізу утворених кластерів, можна сказати, що 2-ох мірні комбінації ніколи не утворюють базиси кластерів, таким чином можемо зменшити навантаженість на алгоритм, зменшивши розмірність комбінацій. Схожа ситуація з великою розмірністю, проте даний момент потребує кількісної оцінки і детального аналізу. Про це детально в розділі 2.3.1.

Процес збереження комбінацій в структуру даних Combination Lexicon дуже схожий до процесу збереження токенів в Token Lexicon. Після побудови всіх комбінацій, потрібних нам розмірностей відбувається почергова перевірка їх наявності в Combination Lexicon. У випадку, якщо структура даних містить указник на комбінацію, відбувається збільшення частоти відповідної комбінації. В противному випадку, в структуру додається посилання на нову комбінацію із частотою 1, генерується відповідна сигнатура. Крім того, існує ще один параметр, який ми будемо використовувати в нашому алгоритмі, це відстань. А саме відстань комбінації від початку назви. Даний атрибут враховується під час визначення пріоритетної комбінації продукта, де таким чином алгоритм кластеризації керується гіпотезою про важливість слів, які стоять на початку тексту, про це в розділі 2.4 [3, с. 11-12].

2.3.1 Оцінка складності структуризації інформації

Лема 1. Оцінка складності структуризації інформації. В найгіршому випадку: (1) $O(n_p(1 + l_{max}) + n_c)$ вставок в структуру даних Forward Index R; $O(n_p * l_{max})$ вставок та пошуків в Token Lexicon L; $O(n_c)$ вставок та пошуків в Combination Lexicon; обчислення відстаней займає $O(n_c)$; генерація всіх k-комбінацій займає (2) $O(n_c * K * log K)$; в той самий час, простір потрібний на обрахування: (3) $O(n_p(1 + l_{max}) + n_c)$. Де n_p означає кількість продуктів, $n_c = O(n_p * N_c(l_{max}, K))$ - це кількість всіх k −комбінацій, а l_{max} означає максимальну кількість токенів в межах одного продукта.

Доведення. В найгіршому випадку кількість токенів для однієї назви є l_{max}. Відповідно, для всієї вибірки, максимальна кількість токенів - $n_p * l_{max}$.

1. Алгоритм (1) виконує $n_p * l_{max}$ пошуків в L_p(множина всі продуктів) та $n_p * l_{max}$ вставок. По схожому принципу, він вимагає $O(n_c)$ на пошуки в L_c та $O(n_c)$. Якщо говорити про Forward Index, алгоритм виконує n_p вставок, відповідно $n_p * l_{max}$ указників до $r_{p,w}$ та n_c до $r_{p,c}$. Саме тому, загальна кількість вставок в R буде $(1 + l_{max}) + n_c$.
2. Побудова всіх k-комбінацій вимагає операції сортування токенів та генерації сигнатури. Беручи до уваги той факт, що назва продукту має Kтокенів, операція сортування вимагатиме $O(K * log\ K)$. Відповідно, для всієї множини комбінацій - $O(n_c * K * log\ K)$.
3. Обрахуємо, кількість простору необхідного для виконання алгоритму. $n_p * l_{max}$, n_c та n_p для зберігання всіх токенів, всіх комбінацій та елементів Forward Index відповідно [3, с. 11].

2.4 Етап 2. Оцінка та пріорітизація комбінацій

Якщо на Етапі 1 ми збирали інформацію, то на Етапі 2, використовуючи дану інформацію ми будемо обраховувати числове значення важливості кожної комбінації. Таким чином можемо порівнювати комбінації між собою за їх мірою важливості. Де комбінація із найбільшим числовим значення міри буде представляти товар, відповідно

утворюватиме домінуючий кластер під час процесу кластеризації. Всі товари, в яких співпадатимуть комбінації із найбільшим числовим значенням будуть належати до одного кластеру. Варто зауважити, що кластери, які ми отримуємо на даному етапі, не є кінцевим результатом нашого алгоритму кластеризації. Тобто після даного етапу кластери будуть модифіковані іншою процедурою.

Головною задачею на даний момент є побудова функції міри важливості комбінації. А саме функції, яка на базі зібраної інформації повертає числове значення кожної комбінації. Параметри, які дана функція враховує при обчисленні:

- *Частота*. Число продуктів, що містить відповідну комбінації. Даний параметр відображатиме вживаність даної комбінації токенів, що свідчитиме про важливість відповідної комбінації;
- *Позиція*. Відстань комбінації від початку назви, яка допоможе перевіряти гіпотезу про важливу інформацію на початку тексту;
- *Довжина*. Кожна комбінація складається із певної кількості токенів. Важливий параметр, за допомогою якого зможемо порівнювати комбінації різної кількості слів;
- *Hot токени*. Як вже зазначалося в розділах 1.2.2, семантика дуже важлива. Правила, описані в Таблиця 1.2.2, якими ми керувалися, коли визначали hot токени, допоможуть пріоритизувати токени між собою.

Побудуємо формули, за допомогою яких будемо оцінювати кожну із характеристик комбінації. Маємо назву товару t, c - комбінацію t та токен $w \in c$. Вважатимемо o_w^c - позиція токена w в комбінації c та o_w^t - позиція токена w в повній назві t. Користуючись цим, відстань комбінації від

повної назви товару ми визначимо за принципом відомої метрики: Евклідова відстань для стрічок:

$$d^2(c,t) = \sum_{w \epsilon c} (o_w^c - o_w^t)^2 \quad (3)$$

Базуючи на рівнянні (3) обрахуємо середню відстань комбінації від початку повної назви товару:

$$\overline{d}(c) = \frac{1}{f_c} \sum_{\forall t:\, c \epsilon t} d(c,t) \quad (4)$$

Отже, на базі (4) рівняння та чотирьох параметрів, які ми описали вище, можемо побудувати функції оцінки комбінації:

$$I(c) = \frac{k * Y_c^2}{\alpha + \overline{d(c)}} * log \quad f_c \quad (5)$$

де $\alpha > 0$ - константа, яка запобігає безкінечності, коли середня відстань комбінації дорівнює нулю, k - кількість токенів в комбінації.

ПараметрY_cвідображає значення IR оцінки. Принцип обрахування значення запозичений із відомого підходу BM25F(Детальніше в 2.4.2).

Концепція, на якій базується алгоритм - це розбиття повної назви товару на віртуальні поля або ж токени. Де кожен токен представляє собою одну із п’яти унікальних семантик, зображених в Таблиця 1.2.2. За схожим підходом BM25F, ми будемо обраховувати Y_c наступним чином:

$$Y_c = \sum_{\forall w \epsilon c} idf(c) \frac{Q(z_{s_w})}{1 - \beta + \frac{\beta * k}{\bar{l}_c}} \quad (6)$$

де $Q(z_{s_w})$ - числова оцінка токена w, залежно від типу семантики. Також, $\bar{l}_c$ - середня відстань всіх комбінацій в датасеті, β- константа, що належить $[0, 1]$, $idf(w)$ - зворотна частота документа, детально описано в підрозділі 2.4.3, виглядає наступним чином:

$$idf(w) = log(\frac{|P|}{f_w}) \quad (7)$$

де $|P|$ - кількість назв товарів, а f_w - частота токена w [3, с. 11-13].

2.4.1 Визначення характеристик кластерів. Первинна кластеризація

Для того, щоб провести класифікацію продуктів, необхідно визначити базис кластерів, а саме характеристики, за якими будемо сортувати продукти. Базис кожного кластеру буде обраховуватися за допомогою вищезазначеної оціночної функції $I(c)$. Отже, для того, щоб визначати домінуючу комбінацію, яка потенційно може стати базисом кластеру, необхідно обрахувати значення оціночної функції для кожної згенерованої комбінації продукта. Варто зауважити, що ми зацікавлені в комбінації, яка має найбільше числове значення оціночної функції. Саме цей факт дозволяє стверджувати, що дана комбінація є базисом кластеру. Крім того, немає потреби зберігати оцінки решти комбінацій, тим самим збільшуючи ефективність пошуку та вставки в наші структури даних.

Детально процес кластеризації виглядає наступним чином. Відбувається ініціалізація пустого списку кластерів U. Під час ітерації всіх продуктів, які належать нашому датасету, ми оцінюємо їх комбінації, визначаючи відстань кожної відносно повної назви за допомогою формули (3), середню відстань всіх комбінацій в межах одного товару на основі (4) формули, де разом з тим, відбувається пошук IR оцінки за підходом BM25F на основі (6) функції. В той же час, в нас є всі необхідні характеристики комбінацій. Отже, ми можемо визначити домінуючу комбінацію за допомогою оціночної функції (5). Домінуюча комбінація буде базисом новоствореного кластеру, який відповідно додається до списку кластерів U у випадку, якщо такого там не знайшлося. В зворотному випадку, відбувається додавання продукту, що аналізується в даний момент часу, до вже існуючого кластеру в U. Отже, на даному етапі ми будемо мати колекцію із сформованих первинних кластерів, які базуються на домінуючій комбінації кожного продукта [3, с. 12-13].

2.4.2 Функція ранжування Okapi BM25F в пошукових системах

Поняття ранжування

Ранжування - процес упорядкування документів або сайтів відповідно до ступеня їх відповідності пошуковому запиту. Головною метою ранжування є розміщення найбільш релевантних документів колекції на вищих позиціях у видачі пошукової системи. Для вирішення завдання ранжування використовуються спеціальні функції, на основі яких і розраховується релевантність [4].

Визначення релевантності

Релевантність є функцією від набору змінних (фактори ранжування). Такими факторами виступають різні числові характеристики документа, за допомогою яких можна розрізняти релевантні документи та нерелевантні. Кількість чинників ранжирування не є фіксованим числом і може змінюватися. Наприклад, Google в даний час при ранжируванні не враховує мета-тег «keywords»(ключові слова) - хоча раніше він мав значення [4].

Функції BM25 и BM25F

Пошукові системи Yandex і Google використовують значно більше таких факторів - функція ранжирування враховує більш ніж 150 компонентів на сьогоднішній день. Велика частина цих факторів є прості числові характеристики документа. Ключовим моментом в ранжируванні є спосіб комбінації параметрів - вид функції релевантності.

В сучасних пошукових системах розрахунок релевантності документів базується на функції Okapi BM25, заснованої на ймовірносній моделі, розробленої в 1970-х і 1980-х роках Стівеном Робертсоном і Карен Спарк Джоунс.

BM25 та його більш сучасні модифікації (наприклад, BM25F) представляють собою TF-IDF-подібні функції ранжирування. TF-IDF (від англ. TF - term frequency, IDF - inverse document frequency) - статистична міра, яка використовується для оцінки важливості слова в контексті документа (що є в свою чергу частиною певної колекції документів). Вага деякого слова пропорційна кількості вживання цього слова в документі, і

обернено пропорційна частоті вживання слова в інших документах колекції.

У реальному веб-пошуку функції ранжирування часто входять як компоненти в набагато більш складну функцію ранжирування. Наприклад, BM25F - модифікація BM25, в якій документ розглядається вже як сукупність кількох полів (заголовки, основний текст, контрольний текст і т.д.), кожному з яких присвоюється своя ступінь значущості в кінцевому вигляді функції ранжування [4].

Принцип роботи алгоритмів серії BM25

BM25 - це пошукова функція, яка впорядковує список документів залежно від наявності або співпадінню ключових слів запиту в документах незалежно від їх близькості. Одне із найпопулярніших представлень цієї функції:

$$score(D,Q) = \sum_{i=1}^{n} idf(q_i) * \frac{tf(q_i, D) * (k_i + 1)}{tf(q_i, D) + k_1 * (1 - b + b * \frac{|D|}{avg\ dl})} \quad (8)$$

де $Q = (q_1, \ldots, q_n)$ - запит, розбитий на токени q_i, D - документ, оцінку якого ми обраховуємо. $tf(q_i, D)$ - TF токена q_i в документі D. $|D|$ - розмірність документа в словах, $avg\ dl$ - середня кількість токенів в документі. В той час, параметри b та k - це константи, які завчасно оптимізовані.

Функція idf у функції $score$ визначається наступним чином:

$$idf(q_i) = ln(\frac{|D| - n(q_i) + 0.5}{n(q_i) + 0.5} + 1) \quad (9)$$

де $|D|$- кількість документів в колекції, $n(q_i)$ - кількість документів, що містять токен q_i[5], [6, с. 347-349].

2.4.3 Алгоритм TF-IDF

Концепція TF-IDF дозволяє оцінити важливість кожного слова чи терміну в документі, який, в свою чергу, є елементом колекції документів. Алгоритм складається із двох етапів: TF та IDF.

TF (англ. term frequency) - частота слова, дозволяє оцінити важливість будь-якого слова в рамках документа. TF - це відношення числа входжень слова до загальної кількості слів у документі. Тобто, простими словами, чим частіше слово зустрічається всередині документа, тим більший умовну вагу має це слово. Функцію можна записати наступним чином:

$$tf(q, d) = \frac{n(q)}{\sum_{t \epsilon d} n(t)} \quad (10)$$

IDF (англ. inverse document frequency) - зворотна частота документа. Іншими словами IDF - інверсія частоти з якою слово зустрічається не в одному документі, а в межах всіх документів певної колекції. Таким

чином, IDF зменшує вагу часто вживаних слів і термінів. Функція виглядає таким чином:

$$idf(t,\ D)\ =\ ln(\frac{|D|}{n(t\ )}) \qquad (11)$$

Отже, алгоритм TF-IDF має кінцевий вигляд [7], [8, с. 4]:

$$tfidf(t, d,\ D)\ =\ tf(t,\ d)\ *\ idf(t,\ D) \qquad (12)$$

2.4.4 Оцінка складності побудови кластерів

Лема 2. Початкове формування кластерів вимагає $O(n_c * l_{max})$ кроків обчислення та $O(|U|\ + n_p)\ =\ O(n_p)$ пошуків та вставок, в найгіршому випадку.

Доведення. Кількість кроків $O(n_c * l_{max})$ необхідна для підрахунку IR оцінок та середніх відстаней кожної k-комбінації з множини розмірністю n_c. Крім того, для кожної назви з n_p продуктів відбувається пошук в множині первинних кластерів U, де загальна кількість пошуків рівна $O(n_p)$. В найгіршому випадку, ці пошуку будуть невдалими, тобто кожен пошук не бучде давати позитивних результатів, саме тому на кожному кроці буде відбуватися операція вставки. Максимальна кількість вставок $O(n_p)$ в множину первинних кластерів [3, с. 14].

2.5 Етап 3. Верифікація кластерів

Оскільки, результат алгоритму кластеризації, який ми отримали після виконання 2-го етапу, допускає порушення умов початкової задачі про unsupervised product matching(UPM), ми потребуємо додаткових операцій - перевизначення кластерів. Головною задачею 3 етапу є реорганізація кластерів на базі певних умов, які були заздалегідь визначені. Такі умови ми будемо називати підтримувальними. Продуктом 3-го етапу будуть кінцеві кластери, будова яких не порушуватиме підтримувальних умов.

2.5.1 Підтримувальні умови

Аналіз показує, що побудова кластерів відбулася з певною похибною: продукти в одному кластері порушують умову задачі.

З метою запобігання подібних випадків визначимо підтримувальні умови, які допоможуть виправити ситуацію: перевизначити кластери. Однією із таких умов є: *один кластер не може містити 2 товари з одного магазину.*

Про актуальність даної підтримувальної умови свідчить наступне доведення від супротивного. Нехай кластер містить 2 продукти від одного магазину. Отже, кластер містить товари, що співпадають один з одним, що означає ідентичність товарів. Що, в свою чергу, означають, каталог магазину містить ідентичні товари. Але це суперечить результатам роботи по очищенню каталогів від дублікатів [3, с. 15].

2.5.2 Визначення зайвих елементів

Покладаючись на підтримувальні умови, ми можемо розпочати етап перебудови кластерів. Очевидним рішенням такої проблеми є визначення елемента в кластері, який є зайвим. І на наступному кроці видалити його із раніше визначеного й розмістити в кластері, який для нього підходить або створити новий, при цьому не порушуючи підтримувальні умови.

Ми маємо виконати валідацію кожного кластера та виявити ті, які порушують підтримувальні умови. Для цього необхідно визначити центроїди всіх кластерів. Центроїдом кластеру будемо називати продукт, який буде мати найбільшу схожість з рештою товарів відповідного кластера. Таким чином центроїд ми будемо розглядати як найкращого представника кластера. Очевидно, центроїд не може бути виключеним із кластера під час процесу верифікації.

Перед тим, як здійснювати пошук зайвих елементів в кластері, нам необхідно відсортувати та згрупувати товари за магазином. Це допоможе нам знаходити магазини, продукти яких порушують підтримувальні умови без перевірки всіх товарів в кластері. Такий підхід дозволить нам позбутися зайвих навантажень на обчислювальні потужності.

Після того, як ми підготували кластери для реструктуризації: відсортували елементи в кластері та визначили центроїди, можемо починати етап перевизначення кластерів. Під час проходу кожного кластера, замість перевірки кожного елемента, ми будемо здійснювати пошук магазину, що порушує підтримувальні умови. У випадку, якщо такий магазин знайдено, ми будемо визначати, який із товарів цього магазину підлягатиме видаленню із поточного кластеру. Для пошуку зайвого елемента магазину, що порушує умови, ми будемо обраховувати коефіцієнт схожості елементів з центроїдом. Відсортувавши елементи від

найменшого до найбільшого числового значення коефіцієнту схожості. Таким чином ми можемо сказати напевно, які елементи даного магазину є зайвими в поточному кластері: всі, окрім товару із найбільшим коефіцієнтом [3, с. 15].

2.5.3 Питання тимчасово видалених продуктів

Нам необхідна тимчасова структура даних DELETED ITEMS CLUSTER (DIC). Вона буде зберігатися товари, причому згруповані за магазином.

Як зазначалося раніше, існує тільки два можливих шляхів вирішення місцерозташування видалених продуктів. Одним із таких шляхів є пошук схожого кластера, в якому додавання товару не порушить підтримувальну умову. В іншому випадку, коли такого кластеру немає, відбувається створення нового, підходящого до нашого продукту.

Варто зауважити, що такий підхід є жадібним, оскільки значну частину ресурсів займає перевірка всіх кластерів й перевірка елемента, який ми порівнюємо з центроїдом кожного кластера.

Саме тому, задля вирішення даної проблеми, ми скористаємося оберненою концепцією: замість пошуку найкращого кластера для кожного елемента із DIC, побудуємо пошук найкращого елемента для кожного кластера. Такий підхід гарантуватиме нам, що кластер отримає найбільш підходящий товар. Такий концепт допоможе уникнути жадібного підходу перебору кожного кластеру для пошуку найкращого для кожного елемента. Саме це пояснює корисність нашої тимчасової структури даних, в яку ми зберігаємо товари задля подальшої роботи з ними.

На наступному кроці головною задачею є розміщення товарів, які на базі вищезгаданого аналізу про підтримувальні умови є неправильно

класифікованими. Для цього ініціалізуємо числове порогове значення схожості t, необхідне для визначення еквівалентних продуктів [3, с. 16].

2.5.4 Послідовна реалізація процесу верифікації

Під час послідовного проходження про вибірці кластерів, перевіримо елементи із DIC, і визначимо найбільш схожий. Процес пошуку найбільш схожого елемента для кластеру базується на обчисленні числового значення за підходу про еквівалентні товари.

Яка дозволяє знайти схожість між елементами із DIC та центроїдом поточного кластера. За допомогою порогового значення t, яке ми визначили вище, зможемо бути впевненими в успішному результаті пошуку найбільш схожого елемента. Нагадаємо, що структура даних DIC містить товари, згруповані за їх магазинами. Відповідно, варто помітити, що обрахування значення схожості відбувається лише з тими магазинами із DIC, елементи яких не порушують підтримувальні умови. Отже, елемент із DIC, який не порушує жодних початкових умов, буде доданий до поточного кластеру, якщо значення функції схожості буде більшим за порогове значення t. У випадку, якщо немає жодного елемента з DIC, значення схожості, якого більше за порогове, стан кластера залишається незмінним - він не поповнюється новим елементом.

Після виконання вищеописаного процесу реорганізації, кластери отримають найбільш схожі елементи. Варто врахувати той факт, що після даного процесу, DIC досі може містити залишкові елементи, оскільки жоден із них не мав значення схожості із центроїдами кластерів більше за порогове.

Отже, залишається задача знайти коректне місцерозташування залишковим елементам із DIC. Цей процес ми побудуємо на наступному принципі: для кожного магазину із DIC і відповідного елементу відбувається пошук найбільш схожий кластер за допомогою підходу, що ми описали вище. У випадку, якщо такого кластера не виявиться - створимо новий, і елемент із DIC перемістимо до нього [3, с. 17].

2.5.5 Поняття еквівалентних товарів

Питання визначення еквівалентних продуктів постала на третьому етапі. Якщо на перших двох етапах процес визначення еквівалентних товарів полягав у визначенні домінантних комбінацій і подальшої класифікації товарів за цією комбінацією. То на третьому етапі означення еквівалентності базується на порівнянні стрічок, тобто повних назв товарів, за допомогою спеціальних метрик, таких як Евклідова відстань, індекс Джакарта, Індекс Соренсена та інші. Задля покращення результатів порівняння двох назв товарів є сенс розглядати комбінацію вищезгаданих метрик. Крім того, вибір такої комбінації є нетривіальною задачею, оскільки подальше використання поняття еквівалентних продуктів є важливим моментом на етапі реструктуризації кластерів, може значно вплинути на кінцевий результат. Отже, підбір остаточної метрики ми визначимо за допомогою емпіричного підходу, а саме методом перебору різних комбінацій метрик на навчальній вибірці продуктів. Варто зауважити, що значення порогової константи є аналітичним, тобто числове значення встановлюється на базі декількох ітерацій алгоритму.

2.5.6 Оцінка складності процесу верифікації кластерів

Підведемо підсумок останнього етапу - процесу верифікації кластерів і відповідно їх реорганізація.

Лема 3. В найгіршому випадку етап верифікації та реструктуризації первинних кластерів потребує $O(n_p^2)$ і також виконує $O(n_p)$операцій із структурами даних.

Доведення. Розберемо детально кожен із підетапів реструктуризації первинних кластерів.

- Процес ініціалізації: $O(n_p^2)$;
- Побудова DIC вимагає $O(n_p * log\, n_p)$;
- $O(|U| * n_p^{DIC})$ на транспортування продуктів з DIC до найбільш підходящих кластерів;
- $O(|U| * n_p^{DIC})$на обробку продуктів, що залишилися після попереднього кроку.

Відповідно, маємо оцінку кожного із підетапів, ми можемо зробити висновок підсумовуючи всі оцінки -

$O(n_p^2 + n_p * log\, n_p + |U| * n_p^{DIC} + |U| * n_p^{DIC}) = O(n_p^2)$.

Таку ж оцінку проведемо для операцій над структурами даних, в найгіршому випадку: $O(n_p^{DIC})$ операцій для видалення та $O(|U| * n_p^{DIC})$

операцій вставки. Отже, загально $O(n_p + n_p^{DIC} + n_p^{DIC}) = O(n_p)$ кількість операцій на структурами даних [3, с. 18].

2.5.7 Підсумкова оцінка складності алгоритму

Теорема. Алгоритм UPM, в найгіршому випадку, виконується за $O(n_p^2 + n_c * (K * log K + l_{max}))$ і $O(n_p * (1 + l_{max}) + n_c)$ потребують операції над структурами даних, при цьому, використовуючи $O(n_p * (1 + l_{max}) + n_c$ простору. Де n_p означає кількість продуктів, $n_c = O(n_p * N_c(l_{max}, K))$ - це кількість всіх k-комбінацій, а l_{max} означає максимальну кількість токенів в межах одного продукта [3, с. 18].

Висновки

Головною метою роботи було дослідження процесу кластеризації товарів різних магазинів, в той же час, від однакових вендорів та побудова відповідної системи прийняття рішень, яка б дозволила класифікувати товари найефективнішим шляхом. Отже, реалізація кожного кроку алгоритма супроводжувалася детальним дослідженням ефективності: оцінки навантаження на систему операціями алгоритму. Разом з тим, під час проектування етапів алгоритма, особливу увагу було присвячено недолікам вже існуючих рішень, а також удосконаленню всіх відомих підходів.

Перший розділ полягав у підготовці даних, первинної обробки даних, а також у визначенні правил, за допомогою яких ми аналізуємо інформацію про продукти та класифікуємо найменші частинки даних - токени. Аналогічно, практична реалізація першого етапу полягає в морфологічному аналізі: побудові семантичних правил та відповідної класифікації токенів.

Основу алгоритму було розкрито в другому розділі. А саме задача полягала у визначенні первинних кластерів. Де означення центроїдів кластерів базувалося на правилах морфологічного аналізу, а також ключових характеристиках назв товарів. Отже, маючи токени та їх комбінації, визначені за певними обмеженнями, таким як кількість слів, порядок токенів, а також статистичні дані про цю інформацію, нам вдалося побудувати числову функцію важливості для оцінки кожної комбінації. Таким чином ми мали змогу порівнювати між собою числове значення важливості кожної комбінації в межах однієї назви товарів. Тобто, тепер кожна назва товару мала свою домінуючу k-розмірну комбінацію токенів, яка найкраще презентувала повну назву продукту. Саме цю найкраще презентуючу комбінацію ми вважаємо базисом первинних кластерів.

Таким чином, всі продукти, що мають однакову домінуючу комбінацію, будуть належати до одного й того ж первинного кластеру. Отже, на другому етапі ми мали вже сформовані кластери, такий результат нас наближував до кінцевої мети - коректна кластеризація продуктів, яка не порушуватиме жодних умов початкової задачі.

Фінальний етап - третій, головною задачею якого було перевірка правильності кластеризації. Відповідно, реструктуризація кожного кластеру, який порушував умови початкової задачі, зокрема підтримувальні умови про неможливу присутність двох та більше товарів від одного магазину в одному кластері. Тобто, головною задачею даного етапу є верифікація та перерозподіл продуктів тих кластерів, які порушують підтримувальні умови. Саме тому, фінальний етап складається з двох підетапів: верифікація та реструктуризація кластерів.

Отже, головним дослідженням цієї роботи було виявити та реалізувати шляхи кластеризації продуктів від різних магазинів, і перевірити коректність та ефективність побудови кінцевих кластерів залежно від початкових підтримувальних умов. В результаті, мета, яка була поставлена - успішно досягнута. Зокрема, проектування алгоритму супроводжувалося практичною реалізацію.

Список використаних джерел

1. Skeels C., Patel Y. Caviar’s Word2Vec Tagging For Menu Item Recommendations. Developer Squareup. URL: https://developer.squareup.com/blog/caviars-word2vec-tagging-for-menu-item-recommendations/ (дата звернення: 25.05.2021).
2. Akritidis L., Bozanis P. Effective Unsupervised Matching of Product Titles with k-Combinations and Permutations. *2018 Innovations in Intelligent Systems and Applications (INISTA)*, м. Thessaloniki, 3–5 лип. 2018 р. 2018. URL: https://doi.org/10.1109/inista.2018.8466294(дата звернення: 24.05.2021). https://ieeexplore.ieee.org/document/8466294
3. A self-verifying clustering approach to unsupervised matching of product titles / L. Akritidis та ін. *Artificial Intelligence Review*. 2020. Т. 53, № 7. С. 4777–4820. URL: https://doi.org/10.1007/s10462-020-09807-8(дата звернення: 24.05.2021).
4. Функции ранжирования в поисковых системах: Okapi BM25, BM25, BM25F. FORTRESS-DESIGN. URL: https://fortress-design.com/bm25/ (дата звернення: 24.05.2021).
5. Contributors to Wikimedia projects. Okapi BM25 - Wikipedia. Wikipedia. URL: https://en.wikipedia.org/wiki/Okapi_BM25 (дата звернення: 24.05.2021).
6. Robertson S., Zaragoza H. Probabilistic Relevance Framework: BM25 and Beyond. Now Publishers, 2009. 70 с.
7. Contributors to Wikimedia projects. TF-IDF – Википедия. Википедия – свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/TF-IDF (дата звернення: 24.05.2021).

8. SPARCK JONES K. A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL. *Journal of Documentation*. 1972. Т. 28, № 1. С. 11–21. URL: https://doi.org/10.1108/eb026526(дата звернення: 24.05.2021).
9. Смерницкий Н. Алгоритм TF-IDF и определение релевантности текста. *Smenik Agency*. URL: https://smenik-agency.ru/blog/algoritm-tf-idf-i-opredelenie-relevantnosti-teksta (дата звернення: 28.05.2021).