УДК 004.9 **doi: 10.26906/SUNZ.2019.2.129**

Є. В. Мелешко

Центральноукраїнський національний технічний університет, Кропивницький, Україна

МЕТОДИ КЛАСТЕРИЗАЦІЇ ГРАФІВ СОЦІАЛЬНИХ МЕРЕЖ ДЛЯ ПОБУДОВИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

Предметом вивчення у статті є процес кластеризації графів соціальних мереж. **Метою** є виявлення методів кластеризації графів соціальних мереж, які можна використати для побудови рекомендаційних систем для соціальних медіа. **Завдання:** провести дослідження існуючих методів кластеризації графів соціальних мереж та дослідити можливість і доцільність їх використання у рекомендаційних системах. Отримані такі **результати:** Проведено дослідження існуючих методів кластеризації графів соціальних мереж двох типів, для одержання кластерів, що не перетинаються, та для одержання кластерів, які можуть перетинатися. Досліджено можливість використання розглянутих методів для побудови рекомендаційних систем соціальних медіа. Досліджено можливості графової СУБД Neo4j по реалізації алгоритмів кластеризації графів. **Висновки.** Було проведено дослідження різних методів кластеризації графів соціальних мереж. Розглянуто методи засновані на оптимізації модулярності графу, на розмітці графу та на методах випадкових блукань, також розглянута окрема група методів, що розбиває граф на кластери, які можуть перетинатися. Досліджено можливість та доцільність використання методів кластеризації графів для побудови рекомендаційних систем. Досліджено можливості графової системи управління базами даних Neo4j для реалізації методів кластеризації графів. Встановлено, що Neo4j надає широкі можливості реалізації розглянутих методів. Для виділення кластерів СУБД Neo4j пропонує декілька реалізованих у її бібліотеці Graph algorithms алгоритмів, а саме Louvain, Label Propagation та Triangle Counting. Проведено тестування функцій бібліотеки Graph algorithms, що реалізують алгоритми Louvain, Label Propagation та Triangle Counting у Neo4j. Інші алгоритми кластеризації графів треба, при необхідності, реалізовувати самостійно, але СУБД Neo4j надає багато зручних інструментів для роботи з даними, які можна використати для реалізації різних алгоритмів кластеризації графів меншими зусиллями, ніж без використання Neo4j.

Ключові слова: кластеризація графів, рекомендаційні системи, соціальні мережі, модулярність, розмітка графів, алгоритми випадкового блукання

Вступ

Одною з основних задач аналізу соціальних мереж (СМ) та соціальних графів є задача виділення співтовариств (кластерів). Співтовариство – це набір вершин, у яких багато спільних ребер між собою та мало спільних ребер з іншими вершинами графу. Вершини одного співтовариства, як правило, мають багато спільних властивостей і саме тому у них виникають спільні зв'язки. Напр., це можуть бути користувачі СМ, що живуть в одному місті, або мають спільні інтереси, або купують схожі товари, тощо. Виділення співтовариств може бути корисним при розробці: рекомендаційних систем, систем виявлення ботів, систем виявлення прихованих соціальних структур, тощо.

Методи пошуку співтовариств за принципом роботи можна поділити на наступні: засновані на оптимізації модулярності, засновані на спектральних особливостях графу та засновані на оцінці ентропії системи [1]. За результатами роботи методи пошуку співтовариств можна поділити на такі, що розбивають граф на кластери, які не перетинаються (Edge Betweenness, Label Propagation, FastGreedy, WalkTrap, Infomap, Leading Eigenvector, MultiLevel, тощо), та на ті, що розбивають граф на кластери, які перетинаються (k-Clique Perlocation, BigCLAM, DEMON, CONGO, тощо) [2].

Метою даної роботи є дослідження методів кластеризації графів соціальних мереж та дослідження можливості і доцільності їх використання для побудови рекомендаційних систем соціальних медіа. Також в ході проведення дослідження було здійснене тестування деяких методів кластеризації графів з застосуванням програмного забезпечення розробленого на мові програмування Python та СУБД Neo4j.

Методи кластеризації графів, засновані на оптимізації модулярності

Переважна більшість алгоритмів кластеризації графів заснована на оптимізації модулярності [2, 3].

Модулярність – деяка числова характеристика, яка описує вираженість структури кластерів в певному графі [1–3]. Для оцінки модулярності можна використовувати формулу:

$$Q = \frac{1}{2n_e}\sum_{ij}\left(A_{ij} - \frac{d_i d_j}{2n_e}\right)\cdot\delta\left(C_i, C_j\right), \qquad (1)$$

де n_e – кількість ребер у графі; A – матриця суміжності графу; d_i – кількість ребер, суміжних з вершиною i; d_j – кількість ребер, суміжних з вершиною j; $\delta(C_i, C_j)$ – дельта-функція, рівна одиниці, якщо $C_i = C_j$ та нулю в іншому випадку.

Дана величина дорівнює різниці між кількістю ребер всередині кластера при поточному розбитті і кількістю ребер, якби вони були випадково згенеровані [3].

Значення модулярності показує вираженість кластерів, вона буде:

– рівна одиниці для повного графу, в якому всі вершини помістили в один кластер;

– рівна нулю для розбиття на кластери, при якому кожна вершина знаходиться в окремому кластері;

– для невдалих розбиттів модулярність може приймати негативне значення.

По-суті, за допомогою значення модулярності можна оцінити якість розбиття графів на кластери. Якісне розбиття має на увазі, що кількість внутрішніх зв'язків всередині кожного кластера має бути більша, ніж кількість його зовнішніх зв'язків.

Значення модулярності характеризує не те, наскільки для даного розбиття внутрішньо-кластерні зв'язки більш щільні, ніж міжкластерні, а те, наскільки вони більш щільні в порівнянні з деякою початковою щільністю. Тому відбувається порівняння з «нульовою гіпотезою», що полягає в тому, що дуги розподілені випадково, тобто немає закономірностей в розподілі щільності дуг всередині графу.

Принцип алгоритмів кластеризації графів, заснованих на оптимізації модулярності, полягає у тому, що: на кожному кроці алгоритму кожній вершині деяким чином ставиться у відповідність деякий кластер, обчислюється значення модулярності та здійснюється перерозподіл вершин графу між кластерами таким чином, щоб збільшити значення модулярності. Робота таких алгоритмів припиняється тоді, коли вже не можна покращити значення модулярності.

Розглянемо деякі найбільш відомі методи кластеризації графу на основі оптимізації модулярності.

Метод FastGreedy – полягає в жадібній оптимізації модулярності [2, 4]. На першому кроці алгоритму кожній вершині графу ставиться у відповідність окремий кластер, а потім об'єднуються такі пари кластерів, об'єднання яких призводить до максимального збільшення модулярності. При цьому об'єднуються тільки інциденті пари вершин, так як, в іншому випадку, модулярність не може збільшитися. Ітерації об'єднання кластерів продовжуються поки продовжує збільшуватися значення модулярності після них.

Метод Louvain (або **Multilevel**) – заснований на багаторівневій оптимізації функції модулярності [2, 5]. Більш якісно розбиває граф на кластертери порівняно з попереднім методом. Даний метод складається з двох частин. Перша частина по суті ідентична методу FastGreedy. Друга частина методу полягає у наступному: створюється новий граф з метавершинами у вигляді знайдених кластерів і ребрами з сумарною вагою всіх ребер, що йдуть від одного кластера до іншого (також створюються петлі з сумарними вагами зв'язків всередині кластера). Такий граф називається метаграф. Алгоритм знову запускається на новому графі. Один з найбільш широковідомих методів за рахунок швидкості роботи.

Метод Leading Eigenvector – спектральний метод, заснований на власних векторах матриці модулярності [2], яка визначається таким способом:

$$B_{ij} = A_{ij} - \frac{d_i d_j}{2n_e}, \qquad (2)$$

де A – матриця суміжності графу; d_i – кількість ребер, суміжних з вершиною i; d_j – кількість ребер, суміжних з вершиною j; n_e – кількість ребер у графі.

Для даної матриці знаходиться перший власний вектор (з максимальним власним числом). Ті вершини, у яких відповідне значення менше нуля, належать одному кластеру, а де більше нуля – іншому. Подібним чином можливий поділ на більшу кількість кластерів.

Дані алгоритми є досить ефективними, але не найпростішими у реалізації.

Методи кластеризації графів, засновані на розмітці графів

Для пошуку кластерів у графі можна використовувати різні методи розмітки та розфарбовування графів.

В основі даних методів лежить ідея, що вершина належить до того кластеру, до якого належить найбільша кількість її сусідніх вершин.

В даних методах дуже важливий порядок перебору вершин.

Методи розмітки графу допомагають перебрати вершини певним чином та з врахуванням їх сусідства визначитися з їх приналежністю до певних кластерів.

Розглянемо найвідоміший метод з даної групи методів – LabelPropagation.

Метод LabelPropagation – розбиває граф на кластери наступним чином: кожна вершина у графі відноситься до того кластеру, якому належить більшість її сусідів, якщо ж таких кластерів декілька, то вибирається випадково один з них [2, 3, 5].

Розглянемо принцип роботи даного методу. У початковий момент часу всім вершинам ставиться у відповідність окремий кластер. Кожна вершина одержує мітку або колір відповідного кластеру. Потім для кожної вершини перевіряється, до яких кластерів належать її сусіди та відбуваються перерозподіл приналежності до кластерів. Через випадковості важливо на кожній ітерації змінювати порядок обходу вершин. Алгоритм закінчує роботу, коли нема чого змінювати: всі вершини відносяться до тих спільнот, що і більшість їх сусідів. Для покращення результатів можна застосувати наступну хитрість – запустити алгоритм декілька разів, кожного разу зберігати результат його роботи та обрати найкращий варіант розбиття графу. Головна перевага даного алгоритму – майже лінійна складність. Недоліком алгоритму є те, що на зашумлених графах часто відбувається об'єднання всіх вершин в одну спільноту, або невелику кількість спільнот.

Методи кластеризації графів, засновані на випадкових блуканнях

Для розбиття графу на кластери можна застосовувати алгоритми випадкового блукання.

Випадкове блукання – математична модель процесу випадкових змін – кроків в дискретні моменти часу. При цьому передбачається, що зміна на кожному кроці не залежить від попередніх змін і від часу.

Розглянемо найвідоміші методи кластеризації графів, засновані на випадкових блуканнях.

Метод Walktrap – використовує ідею про те, що короткі випадкові блукання не призводять до виходу з поточної спільноти (кластеру) [2, 4]. Відстань між вершинами або між групами вершин, роз-

раховують на основі ймовірності досяжності шляху від однієї вершини до іншої в процесі випадкового блукання. Даний показник є великим, якщо вершини знаходяться в різних кластерах в графі, і маленьким, якщо вони знаходяться в одному кластері. Далі здійснюється ієрархічна кластеризація агломеративним способом на основі методу Уорда: вершини об'єднуються в співтовариства на основі вибору найменшого середнього квадрата відстаней між ними. Після того, як вершина приєднана до якого-небудь кластера, відстані між вершинами і кластерами перераховуються.

Метод Infomap – заснований на понятті інформаційних потоків в мережах, кодуванні і стисненні інформації [2]. В даному методі застосовується підхід, що базується на випадкових блуканнях та кодах Хаффмана. У кожної вершини є певна ймовірність її відвідування. За допомогою кодів Хаффмана, відповідно до цих ймовірностей, можна закодувати шлях блукання. Ця послідовність матиме деяку довжину. Якщо використовувати ієрархічне кодування, можна скоротити довжину послідовності. Infomap ґрунтується на жадібному способі мінімізації довжини коду блукання.

Методи розбиття графів на кластери, що перетинаються

Розглянуті вище методи дозволяють розбити граф на кластери, що не перетинаються. В той же час для рішення деяких практичних задач може виникнути необхідність розбити граф на кластери, що можуть перетинатися між собою.

Розглянемо деякі методи, що дозволяють вирішити цю задачу.

Метод Clique perlocation method (CPM) – призначений для розбиття графу на кластери, що перетинаються, використовуючи особливості структури графу, а саме наявності *клік*. Починає роботу з пошуку всіх клік розміру l, після чого будується новий граф, вершинами якого є знайдені кліки [2, 4]. Ребро утворюється в разі, якщо перетин вершин-клік складається з (l– 1) вершин початкового графу. Компоненти зв'язності нового графу і будуть визначати знайдені кластери.

Перевагою методу є його інтуїтивність та простота для розуміння. Недоліком методу є непридатність його використання на графах з дуже великою кількістю вершин.

Слід пояснити термін кліка у теорії графів.

Кліка – підмножина вершин неорієнтованого графу, будь-які дві з яких з'єднані ребром (рис. 1).

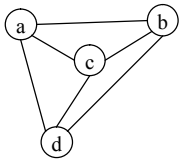

Рис. 1. Приклад кліки у графі

Метод Cluster Affiliation Model for Big Networks – ймовірнісна генеративна модель, що зводить задачу виділення кластерів до задачі факторизації невід'ємних матриць [2, 4]. Застосовується до двудольного графу, в одній частині якого знаходяться кластери, а в іншій – вершини, причому кожна вершина v$\in V$ не просто належить кластеру с $\in C$, а належить йому з деякою невід'ємною вагою. Задача оптимізації полягає у визначенні методом максимізації правдоподоби для даного графу оптимальної матриці зв'язків приналежності до кластерів. Для того, щоб відповісти на питання належить, чи не належить певний об'єкт до певного кластеру, до зв'язків подоби застосовується відсікання по пороговому значенню.

Метод Democratic Estimate of the Modular Organization of a Network (DEMON) – полягає у тому, що для кожної вершини графу будується его-мережа, потім для кожної такої его-мережі окремо застосовується алгоритм Label Propagation, у результаті якого отримуються розбиття на кластери $C_i(u_i)$, для кожного користувача u_i [2]. Усі такі покриття потім об'єднуються в загальне покриття C, яке на початку ініціалізувалося порожнім. Підчас такого об'єднання два кластери об'єднуються в один тільки в тому разі, якщо не більше ε відсотків меншого з них не міститься в більшому з них, наприклад, для $\varepsilon = 0$ об'єднання буде відбуватися тільки, коли один з кластерів повністю міститься в іншому, а для $\varepsilon = 1$ об'єднання буде відбуватися завжди.

Оскільки в реальних системах один і той самий елемент може належати до різних категорій одночасно, то дані методи кластеризації видаються досить корисними для виявлення більш коректного поділу елементів на типи, напр., в рекомендаційних системах.

Застосування методів кластеризації графів у рекомендаційних системах

Оскільки генерація списків рекомендацій у рекомендаційних системах заснована на визначенні схожих між собою користувачів та/або схожих між собою об'єктів (товарів, послуг, контенту) [6, 7], а елементи системи (користувачі, об'єкти) зручно представляти у вигляді графу [8] (рис. 2), то застосування методів кластеризації графів може бути досить доречним та вирішувати питання пошуку схожих елементів.

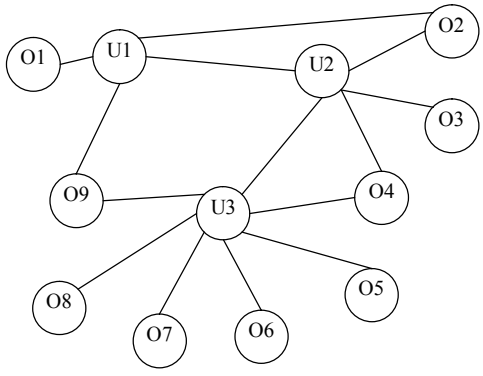

Рис. 2. Приклад графу елементів рекомендаційної системи: вершини типу U – користувачі, вершини типу O – об'єкти системи, ребра – різні взаємодії між елементами системи

Також при застосуванні кластеризації графів у рекомендаційних системах можна враховувати крім зв'язків подоби також зв'язки типу «Дружба» та/або «Підписка». Дані про такі зв'язки легко отримати, якщо рекомендаційна система будується для сайту соціальної мережі, або веб-ресурсу з деякими елементами соціальної мережі, а використання методів кластеризації графів дозволить врахувати ці зв'язки як додатковий параметр у рекомендаційній системі для визначення, до якої категорії віднести елемент системи.

У сучасних рекомендаційних системах для пошуку схожих елементів переважно використовується обчислення коефіцієнтів подоби для кожної пари однотипних елементів системи. Як коефіцієнти подоби можуть виступати коефіцієнт кореляції Пірсона, косинусна відстань, тощо [6, 7]. Такий підхід дозволяє для кожного елемента системи знайти схожі на нього елементи, але не дає можливості розділити елементи на певні категорії та типи. Застосування кластеризації може надати таку можливість та додати гнучкості рекомендаційній системі.

Кластеризувати елементи рекомендаційної системи можна за різними показниками. Так, наприклад, користувачів можна кластеризувати на основі оцінок, які вони ставили об'єктам, на основі властивостей об'єктів, які вони позитивно оцінили, на основі даних з їх профайлів, тощо. Об'єкти можна кластеризувати на основі ключових слів з їх опису, на основі тегів, на основі оцінок, які їм ставили користувачі, тощо.

Після розбиття графу елементів рекомендаційної системи на кластери, можна визначати елементи з якого кластеру/кластерів більше цікавлять певного користувача системи та рекомендувати йому в першу чергу елементи з цього кластеру/кластерів. Також для формування списків рекомендацій певному користувачу можна використовувати вподобання користувачів, які знаходяться з ним в одному кластері.

Для вирішення даної задачі можуть бути цікавими різні методи кластеризації графів, і ті, які розбивають граф на кластери, що не перетинаються, і ті, які розбивають його на кластери, що перетинаються.

Здається більш перспективним розділяти елементи рекомендаційної системи на кластери, що перетинаються, оскільки окремі користувачі можуть мати вподобання в декількох різних сферах, а окремі об'єкти можуть відноситися одразу до декількох категорій.

В той же час, в середині певної сфери інтересів доречно буде додатково поділити користувачів на кластери, що не перетинаються.

Дослідження методів кластеризації графу у даній роботі проводилося для оцінки можливості та доцільності їх застосування у розроблюваному програмному забезпеченні [8] для створення та тестування рекомендаційних систем. Оскільки програмне забезпечення розроблялося на основі графової бази даних Neo4j, доречним буде дослідження наявних у ній методів кластеризації графів.

Засоби для кластеризації графів у графовій СУБД Neo4j

Neo4j – це система управління базами даних типу NoSQL, заснована на представлені даних у вигляді графів [5]. Вона має вбудовану бібліотеку Graph algorithms з розпаралеленими алгоритмами для роботи з графами.

Для кластеризації графів дана бібліотека містить реалізації таких алгоритмів [5]:

– **Louvain** (функція algo.louvain) – алгоритм кластеризації графів, заснований на оптимізації модулярності. Вузли об'єднуються у кластери так, щоб збільшити модулярність. Є одним з найшвидших алгоритмів на основі модульності, і добре працює з великими графами.

– **Label Propagation** (функція algo.labelPropagation) – кластеризує граф, використовуючи лише його структуру. Кожна вершина в графі поміщається в той кластер, якому належить більшість його сусідів. Якщо ж таких кластерів декілька, то вибирається випадково одне з них. У початковий момент часу всім вершинам ставиться у відповідність окреме співтовариство.

– **Triangle Counting / Clustering Coefficient** (функція algo.triangleCount) – визначає кількість трикутників, що проходять через кожен вузол у графі. Трикутник являє собою набір з трьох вузлів, в якому кожен вузол має зв'яки з усіма іншими вузлами. На основі одержаних даних визначає коефіцієнт кластеризації. Хоча розробники СУБД Neo4j відносять даний алгоритм до алгоритмів кластеризації, слід зазначити, що це не зовсім коректно, адже знаходяться трикутники, а не кластери у графі, а трикутник лише частковий випадок кластера.

Також серед реалізацій алгоритмів кластеризації графів у документації до бібліотеки Graph algorithms Neo4j [5] пропонуються до використання **Connected Components** та **Strongly Connected Components** які знаходять зв'язані підграфи незв'язного графу для неорінтованих та орієнтованих графів відповідно, та **Balanced Triads** – алгоритм оцінки структурного балансу графу соціальної мережі, що знаходить збалансовані та незбалансовані тріади у мережі. Оскільки ці алгоритми не розбивають граф на кластери, а виконують трохи інші функції, їх тестування у рамках даного дослідження не проводилося.

Для реалізації інших алгоритмів виділення співтовариств у СМ необхідно розробляти власні функції, існуючі запити Neo4j дають таку можливість.

У розроблюваній системі для роботи з Neo4j була використана бібліотека neo4j.v1 для мови Python. Для підключення до Neo4j використовується наступний код:

```
driver = GraphDatabase.driver
("bolt://localhost", auth=basic_auth(user
= <login>, password = <password>))
session = driver.session()
```

Для здійснення запитів до бази даних використовується функція:

```
session.run("""<запити_до_бази
даних>""", [<змінні_через_кому>])
```

Запис у базу даних інформації можна здійснити за допомогою таких запитів:

```
//створення вузлів
MERGE (U1:User {id:$id1, name:$name1})
MERGE (U2:User {id:$id2, name:$name2})
// створення ребер
MERGE (U1)-[:Friend]->(U2)
```

Дані в Neo4j зберігаються у вигляді на рис. 3.

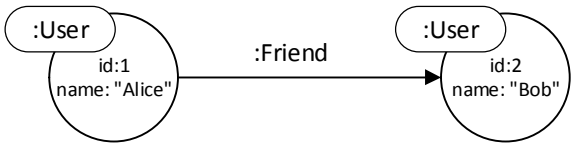

Рис. 3. Формат запису даних у СУБД Neo4j

Для тестування алгоритмів кластеризації графів було згенеровано випадковий граф з властивостями схожими на властивості графів СМ.

Наведемо приклад запиту для виділення співтовариств методом Louvain:

```
CALL algo.louvain('User', 'Friend',
{write:true, writeProperty:'community'})
YIELD nodes, communityCount, iterations,
loadMillis, computeMillis, writeMillis;
```

Після виконання даного запиту у кожної вершини графу з міткою :User з'явиться властивість community, що буде містити номер кластеру, до якого метод Louvain віднесе відповідну вершину.

Результати роботи даного методу для графу з 2348 вузлами (користувачами) та 10762 зв'язками (відношеннями «друзі») наведені у табл. 1.

Запит до бази даних для виконання алгоритму кластеризації Label Propagation:

```
CALL algo.labelPropagation('User',
'Friend','BOTH', {iterations:10,
partitionProperty:'partition', w-
rite:true})
YIELD nodes, iterations, loadMillis,
computeMillis, writeMillis, write,
partitionProperty;
```

Після виконання даного запиту у кожної вершини графу з міткою :User з'явиться властивість partition, що буде містити номер кластеру, до якого метод Label Propagation віднесе відповідну вершину.

Результати роботи даного методу для графу з 2348 вузлами (користувачами) та 10762 зв'язками (відношеннями «друзі») наведені у табл. 2.

Запит до бази даних для виконання алгоритму кластеризації Triangle Counting:

```
CALL algo.triangleCount('User', 'Friend',
{concurrency:4, write:true, write-
?roperty:'triangles',clusteringCoefficient
Property:'coefficient'})
YIELD loadMillis, computeMillis, write-
?illis, nodeCount, triangleCount,
averageClusteringCoefficient;
```

Після виконання даного запиту у кожної вершини графу з міткою :User з'явиться властивість triangles, що буде містити номер трикутника, у якому знаходиться вершина.

Результати роботи даного методу для графу з 2348 вузлами (користувачами) та 10762 зв'язками (відношеннями «друзі») наведені у табл. 3.

Таблиця 1 – **Приклад результатів виклику функції Louvain для графу з 2348 вузлами та 10762 зв'язками**

Кількість кластерів	Кількість ітерацій	Час завантаження даних, мс	Час роботи алгоритму, мс	Час запису результатів у граф, мс
11	2	18	34	93

Таблиця 2 – **Приклад результатів виклику функції labelPropagation для графу з 2348 вузлами та 10762 зв'язками**

Кількість кластерів	Кількість ітерацій	Час завантаження даних, мс	Час запуску алгоритму, мс	Час запису результатів у граф, мс
3	3	25	1	3

Таблиця 3 – **Приклад результатів виклику функції triangleCount для графу з 2348 вузлами та 10762 зв'язками**

Кількість трикутників	Середній коефіцієнт кластеризації	Час завантаження даних, мс	Час роботи алгоритму, мс	Час запису результатів у граф, мс
10630	0.1756	18	3	7

Висновки

Було проведено дослідження методів кластеризації графів. Розглянуто методи засновані на оптимізації модулярності, на розмітці графу та на випадкових блуканнях, також розглянута окрема група методів, що розбиває граф на кластери, що можуть перетинатися. Досліджено можливість та доцільність використання методів кластеризації графів для побудови рекомендаційних систем.

Досліджено можливості графової СУБД Neo4j для реалізації методів кластеризації графів. Neo4j надає широкі можливості реалізації даних методів. Для виділення кластерів вона пропонує декілька реалізованих у її бібліотеці Graph algorithms алгоритмів, а саме Louvain, Label Propagation та Triangle Counting. Інші алгоритми кластеризації графів треба реалізовувати самостійно, але Neo4j надає багато зручних інструментів для роботи з даними, які можна використати для реалізації методів кластеризації графів меншими зусиллями, ніж без використання Neo4j.

Наступні дослідження будуть спрямовані на інтеграцію розглянутих методів кластеризації графів у розроблюване програмне забезпечення для побудови та тестування рекомендаційних систем.

СПИСОК ЛІТЕРАТУРИ

1. Выделение сообществ в графе взаимодействующих объектов / М.И. Коломейченко, И.В. Поляков, А.А. Чеповский, А.М. Чеповский // Фундаментальная и прикладная математика. – Т. 21. № 3. – 2016. – С. 131–139.

2. Никишин Е.С. Методы выделения сообществ в социальных графах [Електронний ресурс] / Е.С. Никишин. – 2016. – Режим доступу: http://www.machinelearning.ru/ wiki/images/8/8a/Nikishin_coursework_community_detection.pdf.
3. Форман Д. Много цифр. Анализ больших данных при помощи Excel / Джон Форман. – М.: Альпина Паблишер, 2016. – 464 с.
4. Пархоменко П.А., Григорьев А.А., Астраханцев Н.А. Обзор и экспериментальное сравнение методов кластеризации текстов // Труды ИСП РАН. 2017. №2. [Електронний ресурс]. Режим доступу: https://cyberleninka.ru/article/n/obzor-i-eksperimentalnoe-sravnenie-metodov-klasterizatsii-tekstov
5. neo4j [Електронний ресурс]. Режим доступу: https://neo4j.com/.
6. Мелешко Є.В. Дослідження методів побудови рекомендаційних систем в мережі Інтернет / Є.В. Мелешко, Г.С. Семенов, В.Д. Хох. // Збірник наукових праць "Системи управління, навігації та зв'язку". Випуск 1(47). – Полтава: ПНТУ ім. Ю. Кондратюка. – 2018. – С. 131–136.
7. Recommender Systems Handbook / Editors Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor. – 1st edition. – New York, NY, USA: Springer-Verlag New York, Inc., 2010. – 842 с.
8. Мелешко Є.В. Розробка рекомендаційної системи на базі субд neo4j. / Є.В. Мелешко, В.В. Босько, В.А. Резніченко // V Міжнародна науково-практична конференція "Інформаційні технології та взаємодії", 20-21 листопада 2018 року, м. Київ. – 2018. – С. 351–352.

Рецензент: д-р техн. наук, проф. С. Г. Семенов,
Національний технічний університет "ХПІ", Харків
Received (Надійшла) 15.02.2019
Accepted for publication (Прийнята до друку) 20.03.2019

Методы кластеризации графов социальных сетей для построения рекомендательных систем

Е. В. Мелешко

Предметом изучения в статье является процесс кластеризации графов социальных сетей. **Целью** является выявление методов кластеризации графов социальных сетей, которые можно использовать для построения рекомендательных систем для социальных медиа. **Задача:** провести исследование существующих методов кластеризации графов социальных сетей и исследовать возможность и целесообразность их использования в рекомендательных системах. Получены следующие **результаты:** Проведено исследование существующих методов кластеризации графов социальных сетей двух типов, для получения кластеров, которые не пересекаются, и для получения кластеров, которые могут пересекаться. Исследована возможность использовать рассмотренные методы для построения рекомендательных систем социальных медиа. Исследованы возможности графовой СУБД Neo4j по реализации алгоритмов кластеризации графов. **Выводы.** Было проведено исследование различных методов кластеризации графов социальных сетей. Рассмотрены методы основанные на оптимизации модулярности графа, на разметке графа и на методах случайных блужданий, также рассмотрена отдельная группа методов, которые разбивают граф на кластеры, которые могут пересекаться. Исследована возможность и целесообразность использования методов кластеризации графов для построения рекомендательных систем. Исследованы возможности графовой системы управления базами данных Neo4j для реализации методов кластеризации графов. Установлено, что Neo4j предоставляет широкие возможности реализации рассмотренных методов. Для выделения кластеров СУБД Neo4j предлагает несколько реализованных в ее библиотеке Graph algorithms алгоритмов, а именно алгоритмы Louvain, Label Propagation и Triangle Counting. Проведено тестирование функций, реализующих алгоритмы Louvain, Label Propagation и Triangle Counting в Neo4j. Другие алгоритмы кластеризации графов нужно, при необходимости, реализовывать самостоятельно, но СУБД Neo4j предоставляет множество удобных инструментов для работы с данными, которые можно использовать для реализации различных алгоритмов кластеризации графов меньшими усилиями, чем без использования Neo4j.

Ключевые слова: кластеризация графов, рекомендательные системы, социальные сети, модулярнисть, разметка графов, алгоритмы случайного блуждания.

Graph clustering methods in social networks for building recommendation systems

Ye. Meleshko

The **subject matter** of the article is the process of graph clustering in social networks. The **goal** is to investigate the graph clustering in methods social networks that can be used to build recommender systems for social media. The **tasks** to be solved are: to study the existing graph clustering methods in social network and investigate the possibility and feasibility of using them in recommender systems. The following **results** were obtained: The study of existing graph clustering methods in social networks of two types, to obtain clusters that do not intersect, and to obtain clusters that can intersect was conducted. The possibility of using the considered methods for building recommendatory systems of social media is investigated. The possibilities of the graph DBMS Neo4j in the implementation of graph clustering algorithms are investigated. **Conclusions**. The study was conducted on various graph clustering methods in social networks. Methods based on optimization of the graph modularity, on the graph labeling and on the methods of random walks and a separate group of methods that find on a graph into intersecting clusters are considered. The possibility and feasibility of using graph clustering methods for constructing recommender systems has been investigated. The possibilities of the graph database management system Neo4j for the implementation of graph clustering methods are investigated. It has been established that Neo4j provides wide possibilities for the implementation of the considered methods. In order to define clusters in a graph, the DBMS Neo4j offers several algorithms implemented in its Graph Algorithms library, namely the Louvain, Label Propagation and Triangle Counting algorithms. The functions that implement the Louvain, Label Propagation and Triangle Counting algorithms in Neo4j were tested. Other graph clustering algorithms need, if necessary, to be implemented on their own, but the DBMS Neo4j provides many convenient tools for working with data that can be used to implement various graph clustering algorithms with less effort than without using Neo4j.

Keywords: graph clustering, recommendation systems, social networks, modularity, graph labeling, random walk algorithms.