Я.П. Башуев, В.Р. Григорьев

Методы деанонимизации в социальных сетях

Работа посвящена исследованию важнейшего направления анализа социальных сетей – разработке методов деанонимизации акторов этих сетей. Целью работы является проведение сравнительного анализа существующих методов и моделей деанонимизации пользователей и разработка модифицированного алгоритма деанонимизации на основе предложенной методики объединения вершин социального графа. Показано, что использование процедуры объединений вершин в графе позволяет осуществить эффективное разделение задачи на эквивалентные подзадачи и тем самым добиться резкого сокращения размерности проводимых вычислений.

Ключевые слова: социальные сети, деанонимизация скрытых пользователей.

Введение

В последние 15 лет происходит беспрецедентный рост информационно-коммуникационных технологий, и в первую очередь социальных сервисов сети Интернет. Плотность пользователей Интернета увеличилась почти в семь раз в период с 2000 до 2015 г. – с 6,5 до 43 процентов мирового населения[1]. Россия в настоящее время занимает 3-е место по объему генерируемого трафика в мире после США и Великобритании[2].

Важнейшей составляющей всего информационного пространства сети Интернет являются социальные сети (СС). По данным компании SimilarWeb, популярные социальные сервисы, такие как Facebook, VK и Google+, занимают лидирующие позиции по популярности среди других веб-сервисов[3]. На рис. 1 представлены данные по развитию основных социальных сервисов, изложенные в докладе директора по технологиям ЦРУ Айра Гас Хант (Ira Gus Hunt) о своем видении роли Big Data на службе ЦРУ, а также возникающих при этом задачах и методах их решения (конференция GigaOM Structure: Data 2013, 20 марта 2013 г., Нью-Йорк)[4].

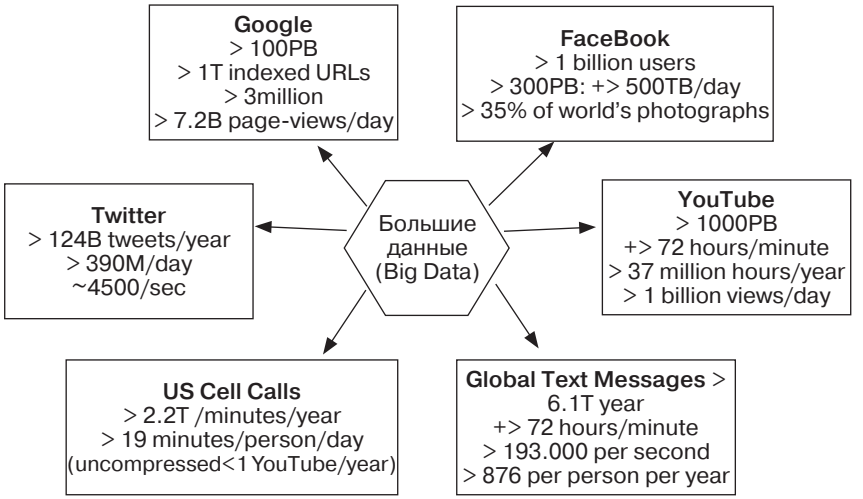


Рис. 1. Основные характеристики социальных телекоммуникаций, составляющих основу «больших данных» (Big Data)

Социальные сети предоставляют своим акторам технологическую платформу для общения и обмена информацией. Пользователи социальных сетей используют их для поиска новых знакомств, коммерческой деятельности, организации мероприятий. При этом они добровольно предоставляют провайдерам персональную информацию, например, номера мобильных телефонов, дату рождения, сообщают о своих увлечениях и интересах. Очевидно, что такая информация может собираться и использоваться сторонними приложениями для рассылки контекстной рекламы или проведения скрытых социальных исследований. В целях предотвращения использования своей персональной информации третьими лицами и недопущения их идентификации как физического лица некоторые пользователи (акторы) социальных сетей стали использовать различные методы сокрытия своего реального лица, получившие название методов анонимизации. Под анонимизацией понимается предоставление акторами СС ложных персональных данных или даже их удаление.

Однако следует особо отметить, что такого рода анонимная идентификация пользователей СС (их анонимизация) может использоваться третьей стороной для организации акций деструктивного характера, нарушающих конституционные законы стран, где они проводились. Неслучайно такого рода акции стали называть Twitter-революциями. События так называемой арабской весны в 2011 г.

(Тунис, Египет, Бахрейн, Ливия и др.)[5] и на Украине в 2013 г.[6] продемонстрировали разрушительную роль неподвластных национальной юрисдикции социальных ресурсов, которые стали инструментами организации так называемых цветных революций, которые, в свою очередь, «мягким» образом привели к реальной смене правящих в этих странах законно избранных исполнительных органов власти, т. е. попросту говоря привели к реальным переворотам в этих странах и изменению их политического курса на полностью проамериканский. Данные примеры показывают актуальность этой поставленной жизнью задачи реализации процедуры деанонимизации пользователей социальной сети.

1. Анализ существующих подходов к деанонимизации пользователей

1.1. Алгоритмы деанонимизации с использованием особенностей браузера

На настоящий момент существует несколько основных направлений реализации процедуры деанонимизации пользователей. Одним из направлений является идентификация конкретных пользователей. Обычно такие алгоритмы используют особенности браузеров в отображении информации, как, например, представлено в следующей работе[7].

Большинство социальных сетей имеют возможность объединения пользователей в группы, например, Facebook и VKontakte. Каждый пользователь $\upsilon \in V$ является членом n_υ групп, где $n_\upsilon \geq 0$. Будем представлять эту информацию в виде вектора $\Gamma(v) = (\Gamma g(v))_{g \in V}$ такого, что:

$$\Gamma g(v) = \begin{cases} 1, \text{ если } v \text{ является членом группы g} \\ 0, \text{ если } v \text{ не является членом группы g} \end{cases}$$

Историю браузера пользователя будем обозначать $ß_\upsilon$. Веббраузер содержит список страниц, которые недавно посетил пользователь. Каждый раз, когда пользователь посещает страницу ρ, ее URL Φ_ρ добавляется в $ß_\upsilon$. Необходимо учитывать, что записи устаревают. Таким образом, по истечении интервала τ URL, относящийся к странице ρ, удаляется из $ß_\upsilon$.

Предполагается, что существует возможность определить, какие страницы из заданного набора пользователь υ посетил. Таким

образом, нарушитель может вычислить для заданного пользователя υ функцию $\sigma_\upsilon(\Phi_\rho)$, которая определена следующим образом.

$$\sigma_\nu\left(\Phi_\rho\right)=\begin{cases}1, \text{ если } \Phi_\rho \in \beta_\nu \\ 0, \text{ если } \Phi_\rho \notin \beta_\nu\end{cases}.$$

Второе предположение заключается в том, что злоумышленник может получить информацию о членах, представляющих интерес *м* групп для заданной социальной сети *S*. Хотя наличие информации о всех группах необязательно, атака, описанная в работе[8], является более эффективной при увеличении числа известных групп.

Информацию об истории браузера пользователя можно получить за счет особенностей отображения гиперссылок браузерами. В современных браузерах ссылки, которые пользователь посетил (т.е. которые находятся в истории браузера), отображаются другим цветом по сравнению с остальными ссылками[9]. С помощью javascript можно узнать, какие ссылки отображаются по-другому и тем самым вычислить функцию $\sigma_\upsilon(\Phi_\rho)$[10].

Большинство социальных сетей обладают одинаковой базовой структурой. Каждый пользователь имеет в сети профиль, который содержит информацию о нем. Взаимодействие пользователя с социальной сетью осуществляет веб-приложение. Это взаимодействие осуществляется с помощью гиперссылок. В примере, представленном на рис. 2, приведены варианты таких гиперссылок.

```
(1) http://www.facebook.com/home.php?ref=home
(2) http://www.facebook.com/ajax/profile/picture/upload.php?id=[userID]
(3) http://www.facebook.com/group.php?gid=[groupID]&v=info&ref=nf
(4) https://www.xing.com/net/[groupID]/forums
(5) http://www.amazon.com/tag/[groupID]/
(6) http://community.ebay.de/clubstart.htm?clubId=[groupID]
```

Рис. 2. Типы гиперссылок в популярных социальных сетях

Как видно из рис. 2, гиперссылки зачастую содержат информацию о пользователе или всей группе. Таким образом, используя историю браузера, можно вычислить, какие группы посещал пользователь, и получить частичный вектор.

После получения частичного вектора посещенных групп атака злоумышленника может продолжиться по одному из двух путей.

Первый способ является более медленным, но более надежным. Используя информацию о членстве в группах, генерируется список кандидатов *C* из объединения всех членов $\{u\}_k$ групп, для которых $\Gamma_k(v) = 1$, т. е. $C = \cup (u)_k$: $\Gamma_k(v) = 1$. Тогда к этому множеству

применяется базовая атака для каждого элемента множества C, т. е. используется базовая атака для определения, является ли ее цель v одним из пользователей из списка кандидатов C.

Второй способ быстрее: используется список кандидатов, состоящий из пересечения множеств всех $\{u\}_k$ для k, из которых $\Gamma_k(v) = 1$. И снова базовая атака используется для проверки: является ли один из пользователей из C целью атаки.

Для работы данного алгоритма необходимо заставить пользователя перейти по ссылке на страницу, которая соберет необходимую информацию об истории браузера. Этот факт является основным недостатком данного алгоритма. Алгоритм требует сбора большого количества информации о группах пользователей целевой социальной сети, что не всегда представляется возможным. Например, социальная сеть ВКонтакте ограничивает количество доступных запросов в секунду[11].

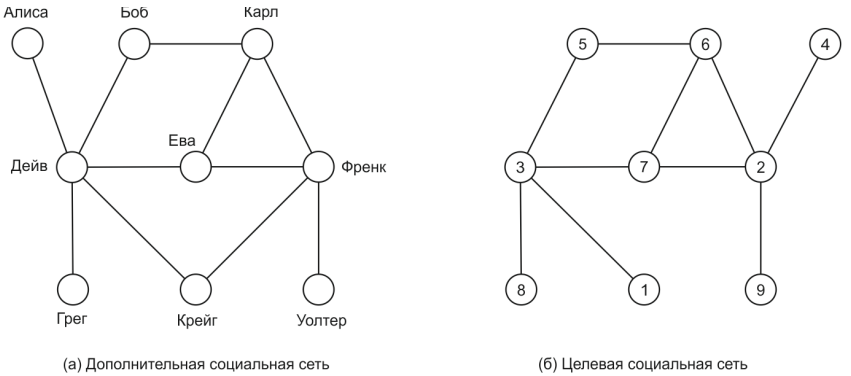
Также большим недостатком этого подхода является невысокая точность предложенного алгоритма. Результатом работы алгоритма является выявление множества пользователей, которые состоят в тех же группах, что и целевой пользователь.

1.2. Алгоритмы деанонимизации с использованием второстепенной социальной сети

В настоящее время одним из основных направлений работ по разработке моделей деанонимизации пользователей социальной сети является разработка алгоритмов, использующих дополнительные данные для идентификации отдельных пользователей, которые извлекаются исходя из анализа других социальных сетей. Данные алгоритмы позволяют провести массовую идентификацию пользователей в заданной социальной сети. Рассмотрим пример, показанный на рис. 3.

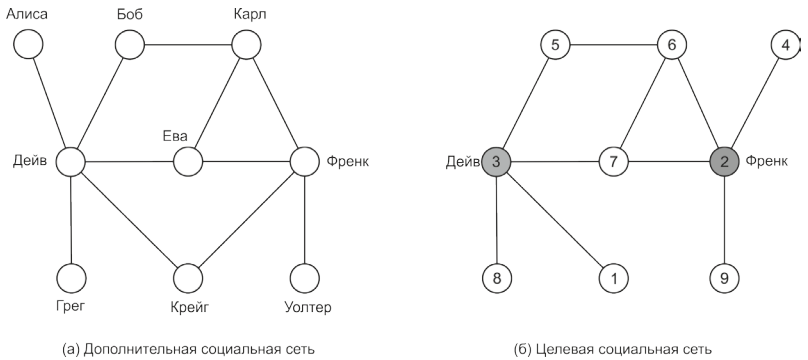
В данном примере необходимо определить имена пользователей на рис. 3б, используя доступную информацию из сети, представленной на рис. 3а. Исходя из имеющейся начальной информации, можно определить, что пользователь 2 – это, с большой вероятностью, Френк, а пользователь 3 – это Дейв, так как они имеют наибольшую степень в данном графе (рис. 4).

Исходя из новой информации, можно определить, что пользователи 6 и 7 это соответственно Карл и Ева, так как их соединения с уже известными вершинами уникальны в данном примере (рис. 5). Таким же образом можно определить, что пользователь 5 это Боб.



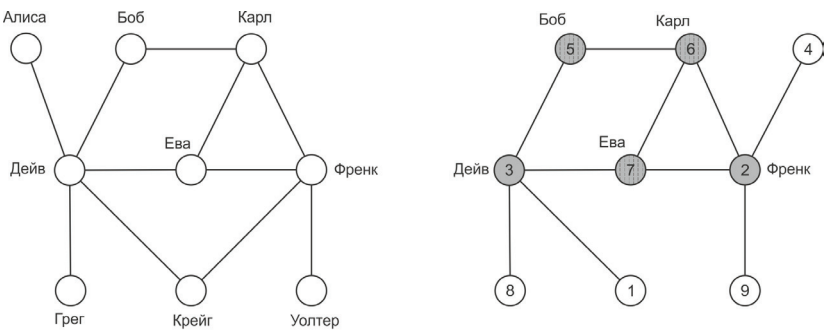
(а) Дополнительная социальная сеть (б) Целевая социальная сеть

Рис. 3. Иллюстрация доступной информации о социальных сетях в виде графов



(а) Дополнительная социальная сеть (б) Целевая социальная сеть

Рис. 4. Идентифицированные пользователи



(а) Дополнительная социальная сеть (б) Целевая социальная сеть

Рис. 5. Идентифицирован второй набор пользователей

Дальнейшая идентификация пользователей с использованием только структурных особенностей графа невозможна, так как оставшиеся вершины не имеют уникальных соединений с остальным графом. В работе[12] разработан алгоритм, который использует атрибуты вершин и ребер графа для нахождения соответствий. Подробности работы алгоритмов с дополнительными данными и модели модифицированной социальной сети рассмотрим на примере работы[13].

Графом социальной сети назовем направленный граф $G = (V, E)$, где V – множество вершин, представляющих пользователей, а E – множество ребер, представляющих отношение пользователей в социальной сети[14]. Элементы множеств V и E имеют наборы атрибутов из множеств X и Y соответственно[15].

Графом $G_{tar} = (V_{tar}, E_{tar})$ назовем граф целевой анонимной социальной сети, а графом $G_{src} = (V_{src}, E_{src})$ назовем граф дополнительной информации (граф дополнительной социальной сети). Очевидно, что для работы алгоритма необходимо, чтобы множество пользователей, представляемых вершинами графов, имело не пустое пересечение. Положим $\tilde{V}_{tar} \subseteq V_{tar}$, $\tilde{V}_{src} \subseteq V_{src}$) множества вершин, соответствующих пользователям, находящимся как в социальной сети G_{tar}, так и в G_{src}. Таким образом, алгоритм, используя входные значения (G_{tar}, G_{src}) и инициализирующее множество отображений μ_0: $V_{src} \to V_{tar}$, находит отображение μ_0: $\tilde{V}_{src} \to \tilde{V}_{tar}$, соответствующее истинному отображению μ_G.

Для определения новых отображений вершин используются различные методы задания веса каждого кандидата на отображение. Например, в работе[16] используется мера близости векторов атрибутов, присущих пользователю социальной сети. В рассматриваемом примере используются структурные свойства графа.

Для каждого отображения определяется вес, который пропорционально зависит от количества уже найденных отображений в соседних вершинах. Очевидно, что отображения, соединяющие вершины с большим количеством уже идентифицированных соседей, имеют большую вероятность оказаться правильными.

Общая схема работы алгоритма:

1. Инициализировать начальные отображения μ_0.
2. Вычислить множество всех соседей V_{nbrs} вершин, для которых отображение известно.
3. Для всех вершин $v \in V_{nbrs}$, используя заданную меру веса, определить лучшего кандидата.
4. Повторять пункты 2–3, пока не перестанут находиться новые отображения.

Эффективность работы алгоритма характеризуется мерой идентификации:

$$R(\mu) = \frac{\sum_{v \in \tilde{V}src} s(v, \mu)}{\left|\tilde{V}_{src}\right|}, \text{ где}$$

$$s(v, \mu) \begin{cases} 0, \nexists \mu(v) \\ 1, \mu(v) = \mu_G(v) \\ -1, \mu(v) = \mu_G(v) \end{cases}$$

Представленное семейство алгоритмов позволяет, при наличии входных данных, провести массовую идентификацию пользователей целевой социальной сети. Точность таких алгоритмов составляет около 75% при условии наличия достаточного количества данных[17,18].

Основными недостатками такого вида алгоритмов являются неочевидность нахождения инициализирующих отображений и их количество. Число начальных отображений, необходимых для эффективной деанонимизации, растет пропорционально размеру сети, что делает идентификацию больших графов затруднительной. Для нахождения отображения для конкретной вершины нужно сравнить атрибуты этой вершины с атрибутами каждой еще не идентифицированной вершины. Это приводит к квадратичному росту времени работы, что является большим недостатком алгоритма в целом.

1.3. Постановка задачи

Как было показано выше, существующие алгоритмы массовой деанонимизации обладают рядом недостатков, которые сильно ухудшают работу этих алгоритмов при больших размерах социальных сетей. На основе анализа существующих алгоритмов были сформулированы следующие задачи:

– создать алгоритм, позволяющий упростить процедуру деанонимизации больших графов социальных сетей;
– улучшить с помощью этого алгоритма существующие подходы к решению задачи деанонимизации;
– создать программное обеспечение, демонстрирующее возможности предложенного подхода.

2. Модель алгоритма деанонимизации пользователей на основе алгоритма построения объединений вершин

Как было показано в предыдущем разделе, существующие алгоритмы деанонимизации используют известные заранее отображения между двумя социальными сетями для расширения известных фрагментов сети на основе сравнения локальных свойств (например, количество и атрибуты идентифицированных соседей) вершин каждого графа. Такие методы требуют большого количества начальных отображений и неустойчивы к большому количеству отличий (например, вершины и ребра, которые есть в одном графе и которых нет в другом). Также с ростом масштаба сети локальные структурные свойства графа дублируются все чаще, делая задачу построения отображений на основе локальных свойств невыполнимой.

В данной работе предлагается модифицированный метод деанонимизации на основе выполнения алгоритма объединения выделенных вершин для дефрагментации постановочной задачи деанонимизации на меньшие подзадачи, решение которых возможно с помощью уже существующих методов деанонимизации.

2.1. Необходимые определения

В целях построения модифицированного алгоритма деанонимизации скрытых пользователей социальных сетей введем следующие определения.

Определение 1. Графом социальной сети назовем ненаправленный граф $G = (V, E)$, где V – множество вершин, представляющих пользователей, а E – множество ребер, представляющих отношение пользователей в социальной сети[19].

Определение 2. Объединением вершин будем называть множество сильно связанных между собой вершин, с минимальным количеством связей с вершинами вне множества[20].

Определение 3. Структурой объединений графа G называется множество $C = \{c_1, c_2, \ldots, c_k\}$, где c_i – объединение вершин, $c_i \neq \emptyset$ и $c_i \cap c_j = \emptyset \ \forall i \neq j, \ i, j \in \overline{1, k}$.

2.2. Алгоритм построения объединений вершин

Для построения объединений вершин графа связей социальной сети в данной работе за основу взят алгоритм InfoMap[21]. Данный алгоритм позволяет быстро найти объединения вершин с высоким качеством разбиения.

Качество разбиения, которое получается при применении таких алгоритмов, измеряется мерой модулярности. Модулярность разбиения – это скалярная величина, лежащая между –1 и 1, которая измеряет плотность соединений внутри множеств разбиения в сравнении с соединениями между множествами разбиения. Также эта мера используется для построения алгоритмов разбиений.

В случае взвешенного графа модулярность определяется формулой:

$$Q = \frac{1}{2m}\sum_{i,j}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j),$$

где A_{ij} – вес ребра между i и j, $k_i = \sum_j A_{ij}$, сумма весов ребер инцидентных i, c_i – объединение, к которому принадлежит i, функция $\delta(u, v) = 1$, если $u = v$ и 0 в других случаях и $m = \frac{1}{2}\sum_{i,j} A_{ij}$.

Алгоритм разделен на два этапа, повторяющихся итеративно. При подготовке к работе алгоритма каждую вершину графа определяют в отдельное объединение, т. е. перед началом работы алгоритма количество объединений равно количеству вершин.

На первом этапе работы алгоритма для каждой вершины i рассматриваются соседи j вершины i и вычисляется изменение модулярности, которое произойдет, если вершину i добавить в объединение, содержащее вершину j. После этого вершина i добавляется в объединение с максимальным изменением модулярности, но только если изменение положительно. Первый этап заканчивается, когда достигнут локальный максимум модулярности, т. е. ни одно перемещение вершины не может дать положительное изменение модулярности. Изменение модулярности ΔQ при переносе вершины i в объединение C для первого этапа легко вычисляется по формуле

$$\Delta Q = \left[\frac{\left(\Sigma_{in} + k_{i,in}\right)}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right],$$

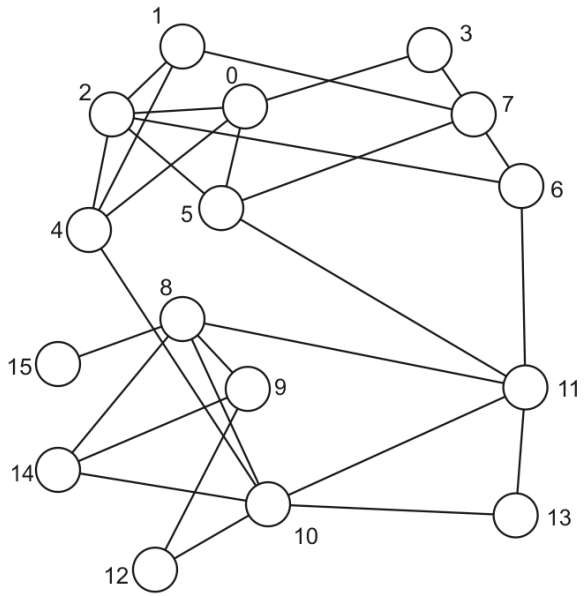


Рис. 6. Граф, в котором нужно выделить объединения

где $\sum_{in}$ – сумма весов ребер внутри C, $\sum_{tot}$ – сумма весов ребер, инцидентных вершинам в C, k_i – сумма весов ребер, инцидентных вершине i, $k_{i,\ in}$ – сумма весов ребер, идущих от i до вершин в C, m – сумма всех весов в графе.

Второй этап алгоритма заключается в построении нового графа, состоящего из вершин, представляющих объединения, найденные на первом этапе. Для этого веса ребер между новыми вершинами задаются суммой весов ребер между вершинами двух объединений. После завершения второго этапа к новому графу можно снова применить данный алгоритм. Эти действия проводятся до тех пор, пока не прекратятся изменения и не будет достигнут максимум модулярности.

Рассмотрим работу алгоритма на примере графа, изображенного на рис. 6.

На первой стадии алгоритма вершины будут распределены по объединениям. Из рисунка видно, что всего объединений 4 (на рис. 7 они окрашены разными цветами).

К графу, изображенному на рисунке 7, применяется второй этап алгоритма, в результате получается второй граф, вершинами которого являются объединения, полученные на первом этапе.

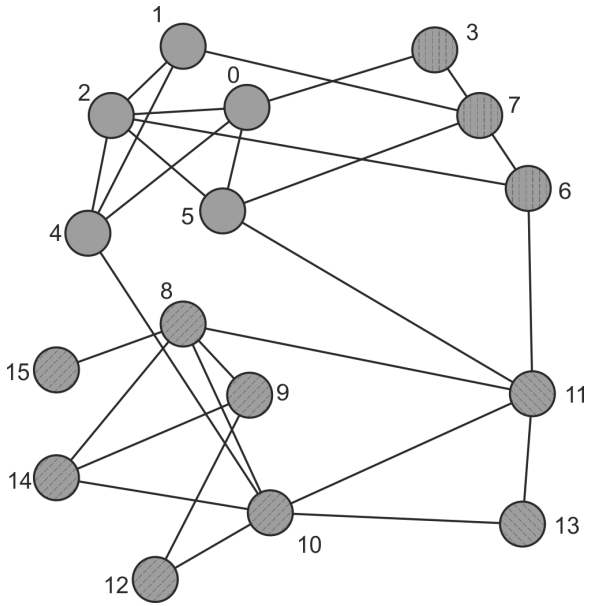


Рис. 7. Граф после первого этапа алгоритма

После первой итерации алгоритма получилось 4 объединения, два из которых имеют значительно меньший вес. Поэтому при применении алгоритма во второй раз останется два объединения большого веса (рис. 9).

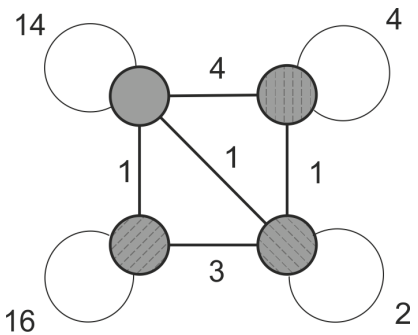


Рис. 8. Результат работы второго этапа реализации алгоритма

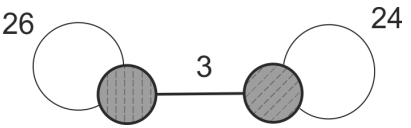


Рис. 9. Вторая итерация алгоритма

2.3. Алгоритм деанонимизации сети на основе реализации процедуры построения объединений

Основным преимуществом использования объединений в графе является эффективное разделение задачи на эквивалентные подзадачи. После применения алгоритма из параграфа 2.2 к имеющимся графам возможно построить отображения объединений между двумя графами и затем начать строить отображения между вершинами графов в каждом отдельно взятом объединении. Для этого процесса возможно также применять существующие алгоритмы деанонимизации[22]. Рисунки 10–14 иллюстрируют последовательность выполнения этапов работы предложенного алгоритма.

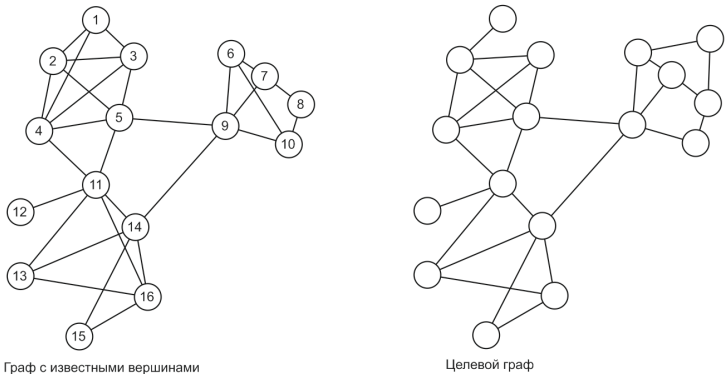


Рис. 10. Начальное состояние графов

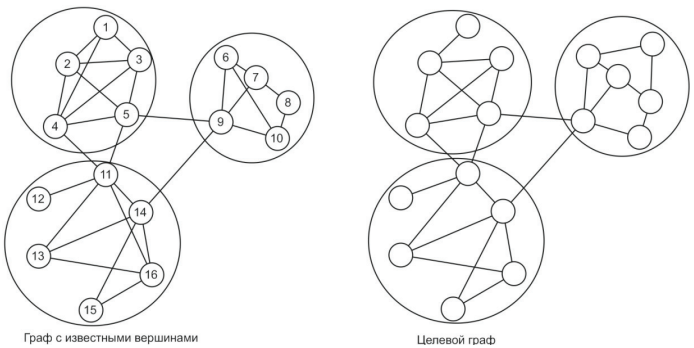


Рис. 11. Построение объединений в графах

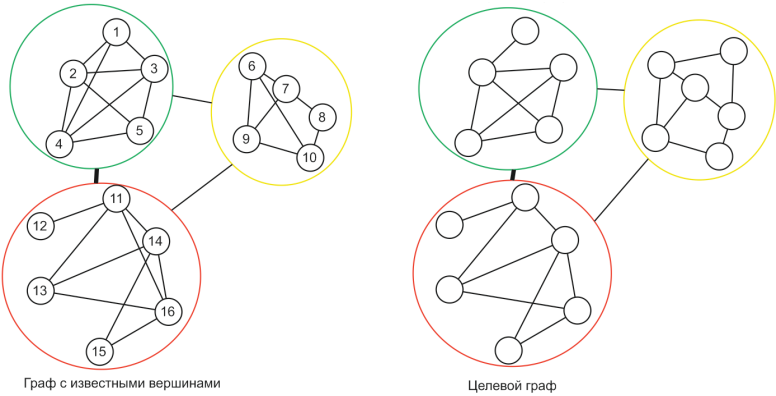


Рис. 12. Построение отображений между графами

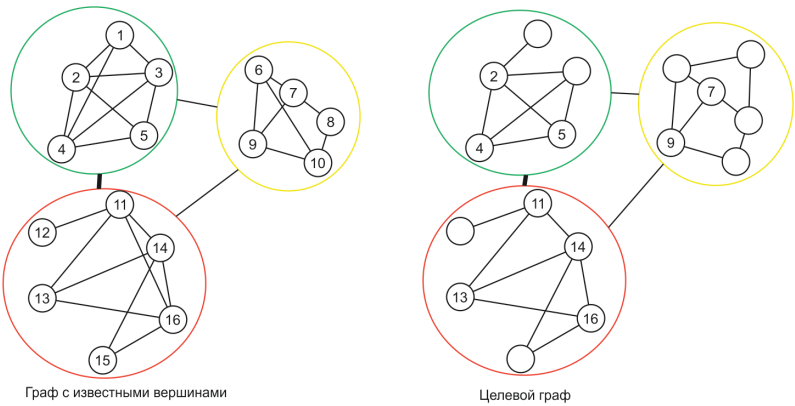


Рис. 13. Идентификация вершин в объединениях

Таким образом, предложенный модифицированный алгоритм состоит из четырех этапов.

1. Построение объединений с помощью InfoMap алгоритма.
2. Нахождение совпадающих пар объединений.
3. Локальная идентификация вершин в отдельных объединениях.
4. Применение алгоритма деанонимизации, не зависящего от числа объединений.

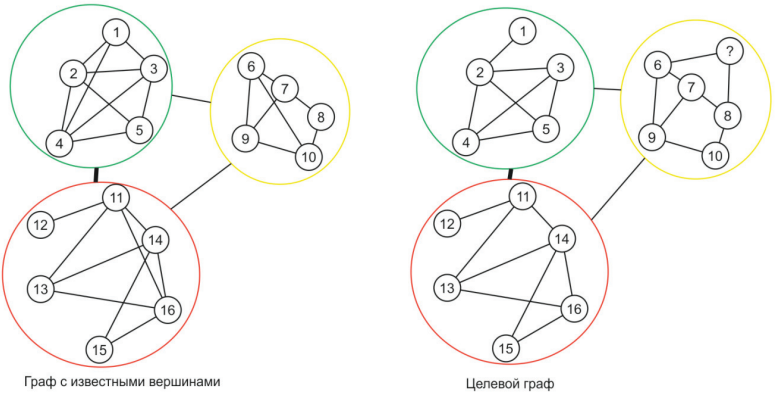


Рис. 14. Результат применения алгоритма деанонимизации

2.4. Отображения объединений

После применения к графам рассматриваемых СС алгоритма InfoMap, необходимо идентифицировать начальные отображения вершин. Затем объединения, которым принадлежат данные вершины, можно отобразить друг в друга. В результате этого процесса могут возникать конфликты, так как вершины в дополнительном графе могут принадлежать разным объединениям, а в основном одному объединению. Для разрешения конфликта идентификации объединения достаточно подсчитать количество идентифицированных вершин для каждого кандидата-объединения и выбрать то, для которого количество вершин максимально.

После того как некоторое количество объединений идентифицировано, строятся высокоуровневые графы объединений. Вершинами данных графов являются объединения, ребрами будут ребра между вершинами, принадлежащими разным объединениям и весом ребер будет количество соединений из одного объединения в другое. Пример построения таких графов показан на рисунках 8 и 9.

К полученным графам применяется модифицированный NS-алгоритм. Графы, полученные из целевой и дополнительной сети, будем обозначать G_1^* и G_2^* соответственно. Множество известных пар вершин будем обозначать M^*.

Как и в оригинальном NS-алгоритме, на каждом шаге алгоритма случайным образом выбирается вершина μ^* из множества сосе-

дей вершин, принадлежащих M^*. Затем для каждой вершины-кандидата из целевого графа вычисляется мера похожести:

$$S(\mu^*, v^*) = \frac{\Sigma(p^*, q^*) \in N(\mu^*, v^*)\left(1 - \left|\sqrt{\omega(\mu^*, \rho^*)} - \sqrt{\omega(q^*, v^*)}\right|\right)}{\sqrt{d(\mu^*)d(v^*)}},$$

где $N(\mu^*, \nu^*)$ – множество идентифицированных соседей вершин μ^* и υ^*, $\omega(\mu^*, \rho^*)$ – вес ребра между вершинами μ^* и ρ^*, $d(\mu^*)$ – степень вершины.

2.5. Локальная идентификация вершин

Полученные в параграфе 2.4 отображения между объединениями позволяют идентифицировать дополнительные вершины с помощью принципа «разделяй и властвуй». Для каждой пары сопоставленных объединений C_1 и C_2 будем рассматривать два подграфа G_{C1} и G_{C2}, содержащих в себе вершины, принадлежащие соответствующим объединениям, и ребра, инцидентные им. Мощность множеств вершин данных графов существенно меньше основного графа, поэтому для каждой вершины можно вычислить локальный коэффициент кластеризации.

Коэффициент кластеризации вершины υ_i определяет, насколько подграф, содержащий эту вершину и все k_i ее соседей, близок к полноте. Коэффициент можно выразить формулой

$$C_{vi} = \frac{2\left|\left(e_{jk} : v_j,\ v_k \in N_i,\ e_{jk} \in E\right)\right|}{k_i(k_i - 1)}$$

Данный параметр и степень вершины можно использовать для попарного сравнения вершин графов G_{C1} и G_{C2} для нахождения отображений. Для сравнения по этим двум параметрам будем использовать две меры близости:

$$D_d(v_i, v_j) = \frac{\left|d(v_i) - d(v_j)\right|}{\max\left(d(v_i), d(v_j)\right)},$$

$$D_{cc}(v_i, v_j) = \frac{\left|C_{v_i} - C_{v_j}\right|}{\max\left(C_{v_i}, C_{v_j}\right)}.$$

Пары вершин, для которых данные меры близки к 1, будем считать вершинами, принадлежащими одному пользователю. Таким образом, будет сопоставлено еще большее количество вершин и, тем самым, будет улучшен результат последнего этапа работы алгоритма.

2.6. Глобальная идентификация вершин

В результате выполнения предыдущих этапов предложенного алгоритма мы получили большое количество отображений из дополнительного графа в целевой. Используя эти отображения в качестве начальных, мы применяем глобальный алгоритм деанонимизации к имеющимся двум графам. В качестве меры похожести для данного этапа будем использовать похожесть идентифицированных соседей. Данный подход в совокупности с большим количеством известных отображений дает простую и быструю меру для получения новых пар[23].

Обозначим целевой анонимный граф как $G_1 = (V_1, E_1)$, а граф дополнительной информации как $G_2 = (V_2, E_2)$. Для каждой вершины u графа G_1 зададим множество $G_m^1(u)$ – множество всех соседей u, для которых найдено отображение в графе G_2. Аналогично для каждой вершины v графа G_2 зададим множество $G_m^2(u)$ – множество всех соседей v, для которых найдено отображение в графе G_1.

Обозначим множество идентифицированных вершин графа как $G_1 : V_S^*, \subseteq V_2$, вершины множества V_S имеют отображения в аналогичное множество графа $G_2 : V_S^* \subseteq V_2$. Определим операцию над этими множествами:

$$N_m^1(u) - N_m^2(v) = N_m^1(u) \setminus \left\{ u_i \in N_m^1(u) : u_i \to v_i \in V_s^*,\ v_i \in N_m^2(v) \right\}.$$

Т. е. операция является разностью множеств с учетом отображения между V_S и V_S^*. С помощью заданной операции можно определить меры различия окрестностей двух вершин[24]:

$$\Delta_1(u,v) = \frac{\left| N_m^1(u) - N_m^2(v) \right|}{\left| N_m^2(u) \right|},$$

$$\Delta_2(u,v) = \frac{\left| N_m^2(u) - N_m^1(v) \right|}{\left| N_m^2(u) \right|}.$$

Полученные меры принадлежат отрезку [0, 1] и равны нулю, если окрестности совпадают. Построим таблицу с количеством строк, равным количеству вершин без пары с соседями в V_i в графе G_1 и с количеством столбцов, равным аналогично количеству вершин без пары с соседями в V_S^* в графе G_2. На пересечении столбца u_i со строкой v_j будет стоять кортеж $(\Delta_1(u_i, v_j), \Delta_2(u_i, v_j))$.

Таблица

Построение таблицы из мер различия окрестностей двух вершин анонимного и дополнительного графов

Δ	u_1	u_2	...
v_1	$(\Delta_1(u_1, v_1), \Delta_2(u_1, v_1))$	$(\Delta_1(u_2, v_1), \Delta_2(u_2, v_1))$	
v_2	$(\Delta_1(u_1, v_2), \Delta_2(u_1, v_2))$	$(\Delta_1(u_2, v_2), \Delta_2(u_2, v_2))$	
...			

Для нахождения подходящего отображения значения обеих мер должны быть минимальными. Поэтому на каждом шаге алгоритма будет выбираться ячейка, для которой значения в кортеже минимальны для всех значений в той же строке и столбце. Выбранные таким образом пары считаются идентифицированными и добавляются во множества. Работа алгоритма продолжается до тех пор, пока не будут найдены все пары.

3. Описание практической части

Практическая часть данной работы была реализована на языке программирования Java 8. Использование данного языка программирования обусловлено следующими причинами:

1) работа имеет в первую очередь исследовательский характер, и никаких специальных требований по возможности интеграции в существующие решения не предъявлялось, в связи с чем выбор языков программирования определялся в основном простотой, гибкостью и скоростью разработки;
2) программы на языке Java исполняются на виртуальной машине Java, что предоставляет возможность запускаться на большинстве современных операционных систем. Также стандарт языка Java содержит в себе большое количество библиотек, что облегчает разработку.

Для работы с графами выбрана библиотека JUNG v2.0.1. Выбор данной библиотеки обусловлен следующими причинами:

1) библиотека JUNG является одной из самых быстрых библиотек по работе с графами и имеет подробную документацию;
2) в состав библиотеки входят средства для визуализации информации, содержащейся в графах;
3) структура библиотеки позволяет модифицировать стандартные возможности под нужды данной работы.

На рис. 15–18 изображены этапы работы алгоритма, предложенного в работе.

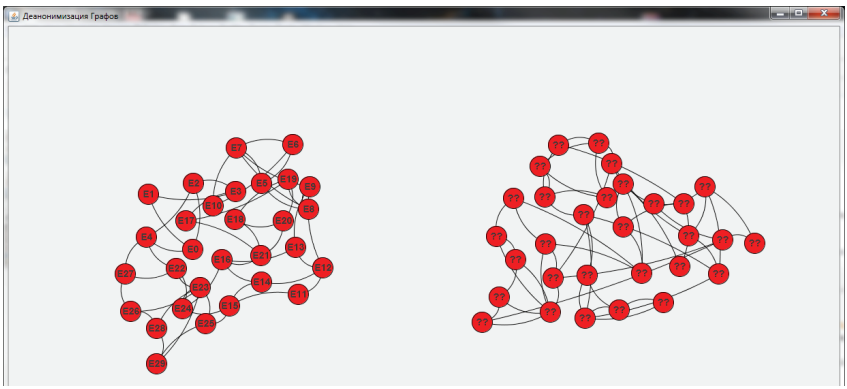


Рис. 15. Начальное состояние.
Слева – граф дополнительной информации,
справа – анонимизированный граф

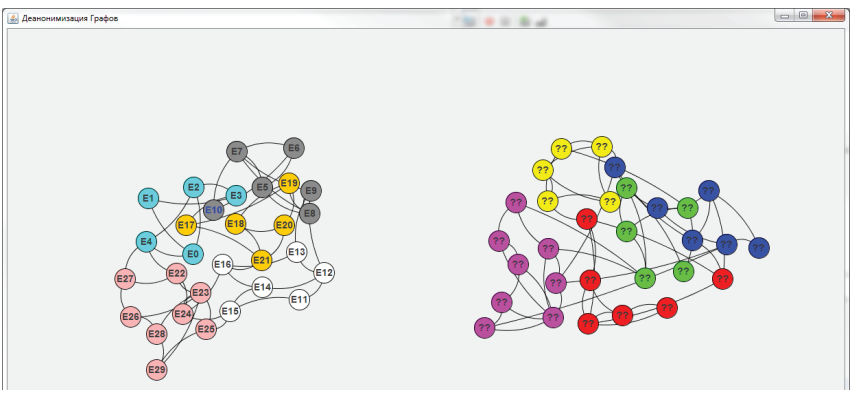


Рис. 16. Состояние программы
после выполнения алгоритма InfoMap

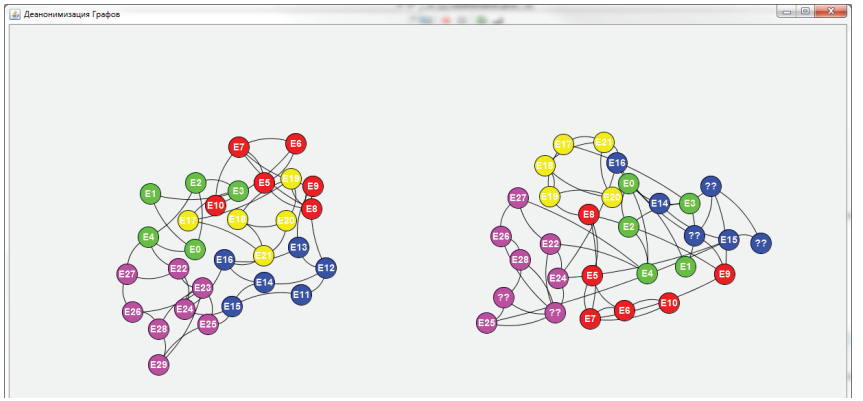


Рис. 17. Построенные отображения разбиений

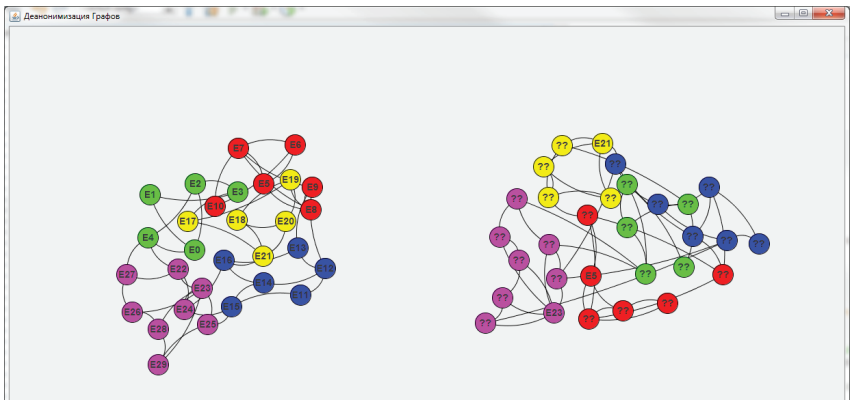


Рис. 18. Результат полной работы алгоритма

Разработанный программный модуль обладает следующими особенностями при практической реализации:

– входные данные и результаты сохраняются в распространенном формате GraphML;
– отображение результатов работы алгоритма осуществляется в графическом формате;
– реализована поддержка различных алгоритмов построения разбиений и поддержка различных методов деанонимизации пользователей социальной сети.

Заключение

В рамках данной работы были получены следующие результаты.

1. Исследованы существующие методы деанонимизации пользователей в социальных сетях.
2. Разработан алгоритм идентификации пользователей с использованием процедуры попарных разбиений.
3. Создан программный модуль, реализующий предложенный алгоритм.

Основной результат данной работы заключается в разработке нового подхода к оптимизации методов идентификации пользователей социальных сетей на основе построения алгоритма попарных разбиений. Предложенный алгоритм позволяет улучшить характеристики существующих реализаций технологии деанонимизации пользователей социальной сети, а также представляет теоретическую и практическую значимость для разработки систем моделирования информационных акций в социальных сетях.

Примечания

[1] Отчет по ИКТ Международного союза электросвязи за 2015 г. [Электронный ресурс] URL: http://www.itu.int/net/pressoffice/press_releases/2015/pdf/17-ru.pdf (дата обращения: 30.08.2016).

[2] Энциклопедия поисковых систем. [Электронный ресурс] URL: http://www.searchengines.ru/seoblog/similar_web_43_mirovogo_t.html (дата обращения: 30.08.2016).

[3] *Голицына А.* Общение заменило поиск // Ведомости. 2015. 11 сент. № 3915. [Электронный ресурс] URL: http://www.vedomosti.ru/technology/articles/2015/09/11/608342-sotsseti-i-messendzheri-oboshli-internet-poisk-i-prosmotr-saitov (дата обращения: 30.08.2016).

[4] ЦРУ – большие задачи и большие данные. На пути к созданию глобального информационного колпака. [Электронный ресурс] URL: https://habrahabr.ru/post/177433/

[5] *Стенин А.* Революция в Египте была раскручена через Facebook // РИА Новости от 12.02.2012. [Электронный ресурс] URL: http://ria.ru/world/20110212/333637995.html (дата обращения: 30.08.2016).

[6] *Алферов К.* Украинская Facebook-революция глазами очевидца // Газета.ru от 24.03.2014. [Электронный ресурс] URL: http://www.gazeta.ru/tech/2014/03/21_e_5959229.shtml (дата обращения: 30.08.2016).

[7] *Wondracek G., Holz T., Kirda E., Kruegel C.* A Practical Attack to De-anonymize Social Network Users: Technical Report TR-iSecLab-0110-001 (2013).

[8] Ibid.

[9] W3C Recommendation. CSS Reference. [Электронный ресурс] URL: http://www.w3.org/TR/CSS21/selector.html%23id-selectors (дата обращения: 30.08.2016).

[10] *Wondracek G., Holz T., Kirda E., Kruegel C.* Op. cit.

[11] Выполнение запросов к API ВКонтакте. [Электронный ресурс] URL: https://vk.com/dev/api_requests (дата обращения: 30.08.2016).

[12] *Narayanan A., Shmatikov V.* De-anonymizing social networks // IEEE Symposium on Security and Privacy. 2009. P. 173–187.

[13] *Simon B., Gulyás G., Imre S.* Analysis of Grasshopper, a Novel Social Network De-anonymization Algotithm // Periodica Polytechnica: Electrical Engineering and Computer Science. 2014. Vol. 58. No. 4. P. 161–173.

[14] Ibid.

[15] *Narayanan A., Shmatikov V.* Op. cit.

[16] Ibid.

[17] *Simon B., Gulyás G., Imre S.* Op. cit.

[18] *Narayanan. A., Shmatikov V.* Op. cit.

[19] *Simon B., Gulyás G., Imre S.* Op. cit.

[20] *Vincent D.* Blondel, Fast unfolding of communities in large networks. 2008.

[21] Ibid.

[22] См., например: *Simon B., Gulyás G., Imre S.* Op. cit; *Narayanan A., Shmatikov V.* Op. cit.

[23] *Wei Peng, Feng Li, Xukai Zou, Jie Wu.* A Two-stage Deanonymization Attack against Anonymized Social Networks // IEEE Transactions on Computers. 2014. Vol. 63. P. 290–303; Feb. 2014, doi:10.1109/TC.2012.202.

[24] Ibid.