МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра прикладной математики**

**КУРСОВАЯ РАБОТА**

**РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА ПО ПРЕДОТВРАЩЕНИЮ ОТТОКА КЛИЕНТОВ ДЛЯ МОБИЛЬНОГО ОПЕРАТОРА НА ОСНОВЕ АНАЛИЗА СОЦИАЛЬНОЙ СЕТИ «ВКОНТАКТЕ»**

Работу выполнил\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_П.С. Куприянов

(подпись, дата) (инициалы, фамилия)

Факультет компьютерных технологий и прикладной математики курс 3

Направление 01.03.02 Прикладная математика и информатика

Научный руководитель

профессор, к.ф. – м.н.,

доцент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Н. Кармазин

(подпись, дата) (инициалы, фамилия)

Нормоконтролер,

к.ф. – м.н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Г.В. Калайдина

(подпись, дата) (инициалы, фамилия)

Краснодар 2018

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc506531390)

[1 Анализ социальной сети 4](#_Toc506531391)

[2 Реализация программы 5](#_Toc506531392)

[2.1 Библиотеки и методы 5](#_Toc506531393)

[2.2 Алгоритм программы 11](#_Toc506531394)

[3 Анализ полученных результатов 16](#_Toc506531395)

[3.1 Анализ по количеству лайков 18](#_Toc506531396)

[3.2 Анализ по количеству репостов 20](#_Toc506531397)

[3.3 Поиск заинтересованных пользователей 22](#_Toc506531398)

[ЗАКЛЮЧЕНИЕ 25](#_Toc506531399)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc506531400)

[ПРИЛОЖЕНИЕ А 27](#_Toc506531401)

# ВВЕДЕНИЕ

Данная работа посвящена анализу данных, представленных в социальной сети «ВКонтакте» – одной из наиболее популярных сетей в России.

Анализ социальных сетей – направление компьютерной социологии, в рамках которой, производится анализ возникающих в ходе социального взаимодействия сетей различной плотности. Результаты, полученные в ходе анализа, могут быть применены в маркетинге в социальных сетях [1].

Целью данной работы является выявление абонентов того или иного мобильного оператора, способных в наибольшей степени влиять на мнение других пользователей социальной сети, а так же абонентов, которые под влиянием чужих мнений вероятнее всего сменят своего нынешнего мобильного оператора. Предполагается, что данная информация поможет оператору выявить возможные пути оттока клиентов с целью их последующего устранения.

В ходе работы была написана программа при помощи языка программирования Python 3.6.3, которая позволяет получить данные о пользователях любого публичного сообщества социальной сети «Вконтакте», а именно: ФИО, контактная информация (если указана), количество друзей, наличие дружеских связей между пользователями социальной сети, PageRank[17], список официальных публичных сообществ мобильных операторов, зарегестрированных в социальной сети, на которые подписан пользователь, количество лайков и репостов записей со страниц официальных публичных сообществ мобильных операторов за указанное время.

Программа использует «API (application programming interface) ВКонтакте» для упрощения создания кода. API представляет собой набор готовых классов, функция и структур для работы с данными, предоставляемыми социальной сетью «ВКонтакте». Это интерфейс, который позволяет получать информацию из базы данных vk.com с помощью http–запросов к специальному серверу. Синтаксис запросов и тип возвращаемых ими данных строго определены на стороне самого сервиса [2].

# Анализ социальной сети

Анализ социальных сетей – это сбор и обработка информации в социальных медиа, необходимый для планирования стратегии взаимодействия с целевой аудиторией. Под взаимодействием в данном случае подразумевается маркетинг в социальной сети.

Анализ социальной сети разумно проводить и как подготовку к старту социального проекта, и как мероприятие по продвижению и сохранению популярности в процессе поддержания проекта.

Анализ социальной сети позволяет оперировать такими данными, как:

* наиболее актуальные темы для целевой аудитории;
* сведения о конкуренции в конкретной области специализации;
* уровень репутации бренда, а так же информация о площадках, которые в той или иной степени на неё влияют.

Анализ социальной сети позволяет:

* более тщательно прорабатывать план работы с целевой аудиторией того или иного бренда;
* своевременно отвечать на реакцию аудитории;

В процессе проделанной работы, была разработана программа, реализующая анализ социальной сети «ВКонтакте» на примере обработки данных со страницы публичного сообщества «Zone23 — страйкбол в Краснодаре», а также сбор данных, на основе которых составляется представление о группах лиц, склонных к изменению своего мнения касательно того или иного мобильного оператора.

# Реализация программы

# Библиотеки и методы

Программа разработана при помощи языка программирования Python 3.6.3. Данное средство разработки – это мощный инструмент для создания программ различного характера. С его помощью можно решать задачи различных типов. Python лёгок в понимании и удобен в использовании, что является несомненным преимуществом данного языка программирования, которое играет немаловажную роль при выборе средства для разработки приложения [3].

В составленной программе используются следующие библиотеки:

* vk – библиотека Python, для работы с инструкциями «API ВКонтакте», которая позволяет отправлять http запросы [4];
* csv – модуль для работы с csv файлами, в том числе для создания и редактирования csv таблиц [5];
* datetime – модуль, предоставляет собой классы для обработки времени и даты [6];
* copy – модуль облегчает операцию копирования объектов языка программирования Python [7];
* pickle – модуль, предоставляющий методы сериализации и десериализации объектов Python [8];

При работе программы производится несколько API запросов с целью получения данных с сайта <https://vk.com/>. В список используемых методов входят:

* execute – универсальный метод, который позволяет запускать последовательность других методов, сохраняя и фильтруя промежуточные результаты. Данный метод позволяет реализовать сбор и анализ большого количества данных в более короткие сроки [9]. Из-за наложения ограничений администрацией социальной сети «Вконтакте», программист может отправлять на сервер не более 3 запросов в секунду. Более того, большинство методов API способно вернуть от 100 до 5000 результатов за один запрос. Таким образом, при анализе большого количества, например, пользователей социальной сети, мы имеем минимальное ограничение обрабатываемых в секунду пользователей, равное 15000. На практике это ограничение намного строже, и часто приходится сталкиваться с максимальным количеством результатов в секунду, равным 300. Метод execute позволяет отправить на сервер vk.com до 25 API обращений за один запрос. Таким образом, программист в теории может отправить до 75 API обращений в секунду. Однако на практике возникают задержки, обусловленные скоростью передачи данных посредством сети Интернет, а так же скоростью работы сервера vk.com, что так же приводит к увеличению времени ожидания результатов;
* wall.get – API метод, предназначенный для работы с записями на стенах сообществ и пользователей сети «ВКонтакте». Метод возвращает список записей со стены. Параметры метода приведены в таблице 1.

Таблица 1 – Список параметров метода wall.get

|  |  |
| --- | --- |
| owner\_id | идентификатор пользователя или сообщества, со стены которого необходимо получить записи (по умолчанию – текущий пользователь) |
| domain | короткий адрес пользователя или сообщества |
| offset | смещение, необходимое для выборки определенного подмножества записей |
| count | количество записей, которое необходимо получить. (максимальное значение: 100) |

Продолжение таблицы 1

|  |  |
| --- | --- |
| filter | определяет, какие типы записей на стене необходимо получить. Возможные значения:  suggests – предложенные записи на стене сообщества (доступно только при вызове с передачей access\_token);  postponed – отложенные записи (доступно только при вызове с передачей access\_token);  owner – записи владельца стены;  others – записи не от владельца стены;  all – все записи на стене (owner + others).  По умолчанию: all. |
| extended | 1 – в ответе будут возвращены дополнительные поля profiles и groups, содержащие информацию о пользователях и сообществах. По умолчанию: 0. |
| fields | писок дополнительных полей для профилей и групп, которые необходимо вернуть. См. описание полей объекта user и описание полей объекта group.  Обратите внимание, этот параметр учитывается только при extended = 1. |

После успешного выполнения возвращает объект, содержащий число результатов в поле count и массив объектов записей на стене в поле items.

Если был задан параметр extended=1, возвращает число результатов в поле count, отдельно массив объектов записей на стене в поле items, пользователей в поле profiles и сообществ в поле groups [10].

* groups.isMember – метод API, возвращает информацию о том, является ли пользователь участником сообщества. Параметры метода представлены в таблице 2.

Таблица 2 – Список параметров метода groups.isMember

|  |  |
| --- | --- |
| group\_id | идентификатор или короткое имя сообщества |
| user\_id | идентификатор пользователя |
| user\_ids | Идентификаторы пользователей, не более 500 |
| extended | 1 – вернуть ответ в расширенной форме.  По умолчанию – 0 |

После успешного выполнения возвращает 1 в случае, если пользователь с идентификатором user\_id является участником сообщества с идентификатором group\_id, иначе 0. 

При использовании параметра extended, возвращает объект, который содержит поля: member (integer, [0,1]) – является ли пользователь участником сообщества, request (integer, [0,1]) – есть ли непринятая заявка от пользователя на вступление в группу (такую заявку можно отозвать методом [groups.leave](https://vk.com/dev/groups.leave)), invitation (integer, [0,1]) – приглашён ли пользователь в группу или встречу.

При передаче нескольких идентификаторов возвращает результат в виде массива объектов, в которых есть поля user\_id (integer) и member (integer, [0,1]) [11].

* likes.getList – метод возвращает  список идентификаторов пользователей, которые добавили заданный объект в свой список Мне нравится. Параметры метода описаны в таблице 3.

Таблица 3 – Список параметров метода likes.getList

|  |  |
| --- | --- |
| type | тип объекта |
| owner\_id | идентификатор владельца Like–объекта: id пользователя, id сообщества (со знаком «минус») или id приложения. Если параметр type равен sitepage, то в качестве owner\_id необходимо передавать id приложения. |

Продолжение таблицы 3

|  |  |
| --- | --- |
| item\_id | идентификатор Like–объекта. Если type равен sitepage, то параметр item\_id может содержать значение параметра page\_id, используемый при инициализации[виджета «Мне нравится»](https://vk.com/dev/Like). |
| page\_url | url страницы, на которой установлен [виджет «Мне нравится»](https://vk.com/dev/Like). Используется вместо параметра item\_id, если при размещении виджета не был указан page\_id. |
| filter | указывает, следует ли вернуть всех пользователей, добавивших объект в список "Мне нравится" или только тех, которые рассказали о нем друзьям. Параметр может принимать следующие значения: likes – возвращать информацию обо всех пользователях;  copies – возвращать информацию только о пользователях, рассказавших об объекте друзьям. По умолчанию возвращается информация обо всех пользователях.  По умолчанию: all. |
| friends\_only | указывает, необходимо ли возвращать только пользователей, которые являются друзьями текущего пользователя. Если метод был вызван без авторизации или параметр не был задан, то считается, что он равен 0. |
| extended | 1 – возвращать расширенную информацию о пользователях и сообществах из списка поставивших отметку «Мне нравится» или сделавших репост. По умолчанию – 0. |
| type | тип объекта |
| offset | смещение, относительно начала списка, для выборки определенного подмножества. Если параметр не задан, то считается, что он равен 0. |
| count | количество возвращаемых идентификаторов пользователей. Если параметр не задан, то считается, что он равен 100, если не задан параметр friends\_only, в противном случае 10. Максимальное значение параметра 1000, если не задан параметр friends\_only, в противном случае 100. |
| skip\_own | не возвращать самого пользователя. |

После успешного выполнения возвращает [список](https://vk.com/dev/list) идентификаторов пользователей с учетом параметров offset и count, которые добавили заданный объект в свой список «Мне нравится».   
Если параметр type равен sitepage, то будет возвращён список пользователей, воспользовавшихся виджетом «Мне нравится» на внешнем сайте. Адрес страницы задаётся при помощи параметра page\_url или item\_id.   
Если extended=1, дополнительно возвращается массив items, содержащий расширенную информацию о [пользователях](https://vk.com/dev/objects/user) или [сообществах](https://vk.com/dev/objects/groups) [12].

* friends.get – метод возвращает список идентификаторов друзей пользователя или расширенную информацию о друзьях пользователя (при использовании параметра fields). Параметры метода приведены в таблице 4.

Таблица 4 – Параметры метода friends.get

|  |  |
| --- | --- |
| user\_id | идентификатор пользователя, для которого необходимо получить список друзей. Если параметр не задан, то считается, что он равен идентификатору текущего пользователя (справедливо для вызова с передачей access\_token). |
| order | порядок, в котором нужно вернуть список друзей. Допустимые значения:  hints – сортировать по рейтингу, аналогично тому, как друзья сортируются в разделе Мои друзья (Это значение доступно только для Standalone–приложений с ключом доступа, полученным по схеме Implicit Flow.).  random – возвращает друзей в случайном порядке.  mobile – возвращает выше тех друзей, у которых установлены мобильные приложения.  name – сортировать по имени. Данный тип сортировки работает медленно, так как сервер будет получать всех друзей а не только указанное количество count. (работает только при переданном параметре fields). По умолчанию список сортируется в порядке возрастания идентификаторов пользователей. |

Продолжение таблицы 4

|  |  |
| --- | --- |
| list\_id | идентификатор списка друзей, полученный методом [friends.getLists](https://vk.com/dev/friends.getLists), друзей из которого необходимо получить. Данный параметр учитывается, только когда параметр user\_id равен идентификатору текущего пользователя. |
| count | количество друзей, которое нужно вернуть. |
| offset | смещение, необходимое для выборки определенного подмножества друзей. |
| fields | список дополнительных полей, которые необходимо вернуть.  Доступные значения: nickname, domain, sex, bdate, city, country, timezone, photo\_50, photo\_100, photo\_200\_orig, has\_mobile, contacts, education, online, relation, last\_seen, status, can\_write\_private\_message, can\_see\_all\_posts, can\_post, universities |
| name\_case | падеж для склонения имени и фамилии пользователя. Возможные значения: именительный – nom, родительный – gen, дательный – dat, винительный – acc, творительный – ins, предложный – abl. По умолчанию nom. |

После успешного выполнения возвращает список идентификаторов (id) друзей пользователя, если параметр fields не использовался.   
При использовании параметра fields возвращает список объектов [пользователей](https://vk.com/dev/objects/user), но не более 5000 [13].

* groups.get – метод возвращает список сообществ указанного пользователя. Параметры метода приведены в таблице 5.

Таблица 5 – Параметры метода groups.get

|  |  |
| --- | --- |
| user\_id | идентификатор пользователя, информацию о сообществах которого требуется получить. |
| extended | если указать в качестве этого параметра 1, то будет возвращена полная информация о группах пользователя. По умолчанию 0. |
| filter | список фильтров сообществ, которые необходимо вернуть, перечисленные через запятую. По умолчанию возвращаются все сообщества пользователя. |

Продолжение таблицы 5

|  |  |
| --- | --- |
| fields | список дополнительных полей, которые необходимо вернуть. Этот параметр учитывается только при extended=1. |
| offset | смещение, необходимое для выборки определённого подмножества сообществ. |
| count | количество сообществ, информацию о которых нужно вернуть. |

После успешного выполнения возвращает объект, содержащий число результатов в поле count (integer) и массив идентификаторов сообщества в поле items ([integer]). Если был задан параметр extended=1, возвращает объект, содержащий число результатов в поле count (integer) и массив объектов, описывающих сообщества, в поле items [14].

* users.get – возвращает расширенную информацию о пользователях. Параметры метода приведены в таблице 6.

Таблица 6 – Параметры метода users.get

|  |  |
| --- | --- |
| user\_id | перечисленные через запятую идентификаторы пользователей или их короткие имена (screen\_name). По умолчанию — идентификатор текущего пользователя. |
| fields | список дополнительных полей профилей, которые необходимо вернуть. |
| name\_case | падеж для склонения имени и фамилии пользователя. Возможные значения: именительный – nom, родительный – gen, дательный – dat, винительный – acc, творительный – ins, предложный – abl. По умолчанию nom. |

После успешного выполнения возвращает массив объектов пользователей[15].

* groups.getMembers – метод возвращает список участников сообщества. Параметры метода приведены в таблице 7.

Таблица 7 – Параметры метода groups.getMembers

|  |  |
| --- | --- |
| group\_id | идентификатор или короткое имя сообщества. |
| sort | сортировка, с которой необходимо вернуть список участников. |
| offset | смещение, необходимое для выборки определенного подмножества участников. По умолчанию 0. |
| count | количество участников сообщества, информацию о которых необходимо получить |
| fields | список дополнительных полей, которые необходимо вернуть. |

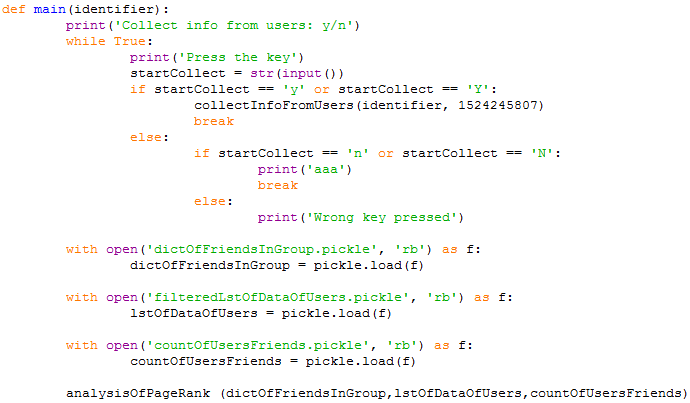
После успешного выполнения возвращает объект, содержащий число результатов в поле count (integer) и массив идентификаторов пользователей в поле items ([integer])[16].

# Алгоритм программы

Для выполнения программы обязательно задать такие параметры, как идентификатор анализируемого сообщества, с помощью которого будет осуществляться доступ к его стене записей, а так же временной порог, вплоть до которого стоит рассматривать записи. В качестве примера был выбран идентификатор группы «Zone23 — страйкбол в Краснодаре». Для определённости будем рассматривать официальные публичные сообщества социальной сети таких операторов, как: «Билайн», «МТС», «Мегафон», «Теле2». Был задан порог времени – 20 апреля 2018 года, что на момент написания работы было эквивалентно одному месяцу. Таким образом, записи со стен групп мобильных операторов, опубликованные ранее двадцатого апреля, рассматриваться в конкретном примере не будут.

Результатом работы программы являются массивы данных, которые конвертируются в csv файлы. Для удобства и наглядности все csv файлы будут импортированы в Excel таблицы.

Основной цикл программы приведён на рисунке 1.



# Рисунок 1 – Основной цикл программы

Полный код программы описан в приложении А.

Основная функция main вызывается от переменной – идентификатор сообщества.

В переменной timeRange хранится дата в формате Unix epoch. UNIX–время или POSIX–время – способ кодирования времени, принятый в UNIX и других POSIX–совместимых операционных системах.

Поскольку в данной программе рассматриваются посты за ограниченный период времени, в этой переменной должна храниться дата, ограничивающая количество рассматриваемых записей со стены группы. Это значение является одним из критериев остановки цикла.

Функция collextInfoFromUsers содержит в себе цикл сбора информации о пользователях, подписанных на рассматриваемое сообщество, а так же о друзьях пользователей. Данная информация выводится в csv файл. Промежуточные данные, такие как списки объектов пользователей отдельно конвертируются в пакеты данных для дальнейшей работы с ними посредством метода pickle [8]. Такой подход позволяет сократить время работы программы, обрабатывая уже имеющиеся данные без дополнительных запросов на сервер vk.com. Получение информации о пользователях рассматриваемой группы происходит в несколько этапов:

* Получение списка идентификаторов пользователей рассматриваемого сообщества посредством вызова функции getListOfMembers;
* Получение необходимой информации о пользователях вызовом функций getInfoOfUsers, getGroupsOfUsersTwo, countOfLikesAndReposts;
* Получение информации о друзьях пользователей посредством вызова функций getFriendsOfUsers, getGroupsOfUsersTwo, FindFriendsInGroups;

Сервер API ВКонтакте имеет частотные ограничения. К методам API ВКонтакте (за исключением методов из секций secure и ads) можно обращаться не чаще 3 раз в секунду. Если превысить частотное ограничение, сервер вернет ошибку. Данное ограничение затрудняет работу при обработке большого количества данных. Для ускорения работы с большими объемами данных был применён метод execute[8], описанный ранее.

Формирование списка пользователей основывается на полученной личной информации а так же на информации о друзьях пользователей. Сформированный список преобразуется в cvs файл для упрощения проведения дальнейшего анализа. Пример формализации данного файла представлен ниже в таблице 8.

Данные формализуются в качестве таблицы. В данном представлении столбцы предоставляют информацию о конкретном пользователе. Строка ID содержит идентификатор страницы пользователя в социальной сети. Строка phone содержит информацию о мобильном телефоне, который указал пользователь на своей странице в социальной сети. В строке operator содержится информация о операторе мобильного телефона, который был указан пользователем. Строка groups содержит список групп мобильных операторов, в которых состоит пользователь. Строки типа lОператор и rОператор содержат информацию об активности пользователя в публичных сообществах мобильных операторов. Они показывают количество лайков и репостов записей со страниц групп за указанный нами промежуток времени соответственно. Строки типа fОператор содержат информацию о количестве друзей пользователя, подписанных на официальную группу того или иного оператора в социальной сети.

Таблица 8 – Результат формирования списка пользователей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | 134868354 | 167887578 | 46508328 | 100349311 |
| phone | +79528520763 | 890086742866 | 8900-230-13-20 | - |
| Operator | теле2 | теле2 | теле2 | - |
| groups | ['теле2'] | ['мтс'] | ['мегафон'] | ['билайн', 'мтс', 'мегафон', 'теле2'] |
| lTele2 | - | - | - | 1 |
| rTele2 | - | - | - | - |
| lMTS | - | - | - | - |
| rMTS | - | - | - | - |
| lMegafon | - | - | - | - |
| rMegafon | - | - | - | - |
| lBeeline | - | - | - | - |
| rBeeline | - | - | - | - |
| fTele2 | 0 | 0 | 1 | 1 |
| fMTS | 0 | 1 | 4 | 2 |
| fMegafon | 1 | 0 | 3 | 3 |
| fBeeline | 0 | 0 | 0 | 0 |

Идентификационные номера пользователей, представленные в таблице, соответствуют идентификационным номерам страниц пользователей социальной сети «ВКонтакте».

Полученные данные уже можно использовать для составления оценки заинтересованности пользователя в услугах того или иного оператора.

Далее программа выполняет анализ сформированного списка посредством вызова функции analysisOfPageRank. Процесс анализа состоит из нескольких этапов:

Фильтрация полученного массива данных путём отбора для дальнейшего анализа только тех пользователей, которые подписаны на ту или иную публичную страницу мобильного оператора.

Вычисление значения PageRank каждого пользователя, а также формирование списка идентификаторов друзей пользователя, которые так же подписаны хотя бы на одну из публичных страниц мобильных операторов. Формирование нового списка пользователей.

Вычисление PageRank друзей пользователей, а также составление списка официальных групп в социальной сети, на которые подписаны друзья пользователей. Формирование сопоставительного списка, где каждому пользователю ставится в соответствие каждый пользователь из списка друзей.

Обновлённый список пользователей и сопоставительный список преобразуются в csv файлы для удобства представления. Примеры таких списков указаны в таблицах 9 и 10.

В таблице 9 каждая строка представляет информацию о конкретном пользователе. Столбец ID содержит идентификатор пользователя в социальной сети. PageRank содержит результат вычисления значения PageRank пользователя. Столбец Groups содержит список групп мобильных операторов, на которые подписан пользователь. Столбец Friends содержит список друзей пользователя, которые подписаны хотя бы на одно из публичных сообществ мобильных операторов.

Таблица 9 – Результат формирования обновлённого списка пользователей

|  |  |  |  |
| --- | --- | --- | --- |
| ID | PageRank | Groups | Friends |
| 9014712 | 0.018958064760749 | ['теле2'] | ['13138925', … , '216506369'] |
| 72537669 | 0.0117729582164252 | ['билайн'] | ['19348416', … , '216073665'] |
| 321445490 | 0.0134412679153711 | ['мтс', 'теле2'] | ['228979612', … , '453282036'] |

Таблица 10 – Результат формирования сопоставительного списка

|  |  |  |  |
| --- | --- | --- | --- |
| Owner | ID | PageRank | Groups |
| 9014712 | 13138925 | 0.00368926093035782 | ['мтс'] |
| 9014712 | 16803272 | 0.00182987342145747 | ['мегафон'] |
| 321445490 | 228979612 | 0.00368926093035782 | ['билайн'] |

В таблице 10 каждая строка представляет нам сопоставление пользователя с одним из его друзей. Столбец Owner содержит идентификаторы пользователей. Столбец ID содержит идентификаторы друзей пользователей соответственно. PageRank содрежит результат вычисления значения PageRank друга. Groups содержит список официальных групп в социальной сети мобильных операторов, на которые подписан друг.

Разработанная программа является универсальной. Её можно применять для анализа данных любого публичного сообщества, изменив основной параметр – идентификатор анализируемого сообщества. Кроме того, путём замены ограничителя по времени можно захватить более широкий, либо более узкий объём данных при сборе информации.

# Анализ полученных результатов

После выполнения программы, был получен список 6075 записей, содержащих информацию о подписчиках сообщества «Zone23 — страйкбол в Краснодаре». Пример списка был продемонстрирован в таблице 8.

Для данного примера были отобраны пользователи, которые указали свой номер мобильного телефона на странице в социальной сети, либо являются подписчиками одного из публичных сообществ мобильных операторов. В действительности, при анализе выбранного нами сообщества, отмеченные данные указывает лишь малый процент пользователей, и именно этих пользователей мы будем анализировать в дальнейшем.

# Анализ первичного списка пользователей

Обратимся к таблице 8, указанной ранее. Обладая представленным набором данных, уже можно сложить поверхностное мнение о заинтересованности того или иного пользователя. Так пользователь с идентификатором 46508328 хотя и является клиентом компании «Теле2», но является подписчиком публичного сообщества оператора-конкурента. Кроме того, некоторые из его друзей так же подписаны на группы других операторов. На практике, при анализе подписчиков случайного сообщества, исходя из 6075 примеров, можно сделать вывод, что лишь малая их часть интересуется новостями, выкладываемыми мобильными операторами в официальных сообществах в социальной сети. Это следует как и из малого количества пользователей, подписавшихся на группы операторов, так и из ещё более малого количества лайков и репостов записей со страниц этих групп. Так из указанного количества проанализированных пользователей лишь 86 человек оказались подписчиками публичного сообщества того или иного мобильного оператора. Полученной на данном этапе информации недостаточно для составления прогноза. Поэтому проведём дальнейший анализ.

# Анализ обновленного и сопоставительного списков.

Обратимся к представленным выше таблицам 9 и 10. Анализ будем проводить сразу по двум представлениям. Разберём пример позьзователя с идентификатором 9014712. Как указано в таблице 9, его PageRank составляет примерно 0.019, он является подписчиком официального сообщества оператора «Теле2». В таблице 10 представлены два идентификатора, владельцы которых являются друзьями пользователя: 13138925, 16803272. Значения PageRank этих пользователей равны примерно 0.004 и 0.002, а их заинтересованность – это МТС и Мегафон соответственно. PageRank – это значение, измеряющее «важность» или «авторитетность» элемента среди остальных. Изначально данный критерий применялся для ранжирования веб-сайтов. Однако он может применяться к любому набору объектов, которые возможно представить в виде графа. Итак, получим, что пользователи с идентификаторами 13138925, 16803272, являясь друзьями пользователя 9014712, попадают под его влияние. Последний же, обладая большей степенью авторитетности, способен изменить мнение двух других пользователей в большей мере.

# ЗАКЛЮЧЕНИЕ

В ходе работы была составлена программа, позволяющая получать информацию из базы данных vk.com, выделять необходимые для анализа поля и записывать их в удобный для дальнейшей работы табличный вид.

На основании полученных данных, был произведён анализ участников публичного сообщества «Zone23 – страйкбол в Краснодаре».

Проанализировав полученные результаты, был составлен список пользователей, которые в той или иной степени подверженных влиянию своих друзей. Используя полученную информацию, оператор может более направлено составлять коммерческие предложения для более узкого круга лиц с целью предотвращения оттока клиентов, либо для увеличения вероятности перехода пользователя от стороннего оператора.

Если правильно применить выводы данного анализа, то они могут помочь в привлечении клиентов, а так же в контроле путей оттока пользователей к конкурентам. Полученная информация призвана показать наиболее актуальные группы людей, для которых стоит составить новое коммерческое предложение, что должно снизить затраты на маркетинговые компании, направленные на более широкую аудиторию, предпочтения которой изначально не известны

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Халилов Д. Маркетинг в социальных сетях: Манн, Иванов и Фербер. Москва. 2013. 240 c.
2. Знакомство с API ВКонтакте URL: <https://vk.com/dev/first_guide> (дата обращения: 15.05.2018)
3. Программирование на Python: Часть 1 URL: [https://www.ibm.com/developerworks/ru/library/l–python\_part\_1/index.html](https://www.ibm.com/developerworks/ru/library/l-python_part_1/index.html) (дата обращения: 15.05.2018)
4. Библиотека vk для работы с VK API URL: <https://habrahabr.ru/post/319178> (дата обращения: 18.05.2018)
5. Reading and Writing CSV Files in Python URL: [http://stackabuse.com/reading–and–writing–csv–files–in–python](http://stackabuse.com/reading-and-writing-csv-files-in-python) (дата обращения: 19.05.2018)
6. Модуль datetime URL: [https://pythonworld.ru/moduli/modul–datetime.html](https://pythonworld.ru/moduli/modul-datetime.html) (дата обращения: 19.05.2018)
7. Модуль copy URL: <https://pythonworld.ru/moduli/modul-copy.html> (дата обращения: 20.05.2018)
8. Модуль pickle URL: <https://pythonworld.ru/moduli/modul-pickle.html> (дата обращения: 20.05.2018)
9. Метод execute URL: <https://vk.com/dev/execute> (дата обращения: 20.05.2018)
10. Метод wall.get URL: <https://vk.com/dev/wall.get> (дата обращения: 20.05.2018)
11. Метод groups.isMember URL: <https://vk.com/dev/groups.isMember> (дата обращения: 21.05.2018)
12. Метод likes.getList URL: <https://vk.com/dev/likes.getList> (дата обращения: 21.05.2018)
13. Метод friends.get URL: <https://vk.com/dev/friends.get> (дата обращения: 21.05.2018)
14. Метод groups.get URL: <https://vk.com/dev/groups.get> (дата обращения: 24.05.2018)
15. Метод users.get URL: <https://vk.com/dev/users.get> (дата обращения: 24.05.2018)
16. Метод groups.isMember URL: <https://vk.com/dev/groups.isMember> (дата обращения: 24.05.2018)
17. PageRank URL: <https://en.wikipedia.org/wiki/PageRank> (дата обращения: 20.05.2018)

# ПРИЛОЖЕНИЕ А

Листинг программы

import time

import copy

import csv

import datetime

import pickle

import login

api = login.api

requestTime = 0

class User:

ID = ''

phone = ''

operator = ''

groups = ''

operators = {

'мтс': [918, 978, 988, 989], 'билайн': [903, 905, 906, 909, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969],

'теле2': [900, 901, 902, 908, 952, 953, 995], 'мегафон': [928, 929, 938, 999]

}

officialGroups = {

'мтс': 8458649, 'билайн': 26514504, 'теле2': 18098621, 'мегафон': 3785

}

listOfOfficialGaroups = [8458649,26514504,18098621,3785]

isSetGroupOperator = {

'мтс' : 0, 'билайн': 0,

'теле2': 0, 'мегафон': 0

}

globalIDForFindUsersList = ''

def req ():

global requestTime

requestTime += 1

if requestTime == 3:

time.sleep(1)

requestTime = 0

def GetUser (user):

listOfNubers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

try:

userId = str(user['id'])

except:

userId = str(0)

try:

if len(user['mobile\_phone']) <= 20:

strPhone = user['mobile\_phone']

countOfNumbers = 0

for s in strPhone:

if s.isdigit():

countOfNumbers += 1

if countOfNumbers >= 10:

userPhone = str(user['mobile\_phone'])

else:

userPhone = str(0)

else:

userPhone = str(0)

except:

userPhone = str(0)

operator = getOperatorOfUser(userPhone)

user = {

'id': userId, 'userPhone': userPhone, 'operator': operator, 'groups': '', 'lTele2' : '', 'lMTS' : '',

'lMegafon' : '', 'lBeeline' : '', 'rTele2' : '', 'rMTS' : '', 'rMegafon' : '', 'rBeeline' : '',

'fTele2' : '', 'fMTS' : '', 'fMegafon' : '', 'fBeeline' : ''

}

return user

def WriteCSVFileOfUsers (name, user, create):

if create is True:

data = ["ID;phone;operator;groups;lTele2;rTele2;lMTS;rMTS;lMegafon;rMegafon;lBeeline;rBeeline;fTele2;fMTS;fMegafon;fBeeline".split(";")]

with open(name, 'w', encoding = "utf-8") as file:

writer = csv.writer(file, delimiter = ';')

for line in data:

writer.writerow((line))

else:

with open(name, 'a', encoding = "utf-8") as file:

writer = csv.writer(file, delimiter = ';')

writer.writerow((user['id'], user['userPhone'], user['operator'], user['groups'], user['lTele2'], user['rTele2'], user['lMTS'], user['rMTS'], user['lMegafon'], user['rMegafon'], user['lBeeline'], user['rBeeline'], user['fTele2'], user['fMTS'], user['fMegafon'], user['fBeeline']))

def writeCSVFileOfAnalysisUsers (name, user, create):

if create is True:

data = ["ID;PageRank;groups;Friends".split(";")]

with open(name, 'w', encoding = "utf-8") as file:

writer = csv.writer(file, delimiter = ';')

for line in data:

writer.writerow((line))

else:

with open(name, 'a', encoding = "utf-8") as file:

writer = csv.writer(file, delimiter = ';')

writer.writerow((user['id'], user['PageRank'], user['groups'], user['friends']))

def writeCSVFileOfAnalysisFriends (name, user, create):

if create is True:

data = ["owner;ID;PageRank;groups".split(";")]

with open(name, 'w', encoding = "utf-8") as file:

writer = csv.writer(file, delimiter = ';')

for line in data:

writer.writerow((line))

else:

with open(name, 'a', encoding = "utf-8") as file:

writer = csv.writer(file, delimiter = ';')

writer.writerow((user['owner'], user['id'], user['PageRank'], user['groups']))

def getListOfMembers (idOfGroup):

offset = 0

request = api.execute.getListOfUsersInGroup(idGroup = idOfGroup, offset = offset, v = 5.73, timeout = 120)

req()

b = request['users']

listOfMembers = []

for i in b:

listOfMembers.append(str(i))

while request['overCount']:

offset += 25000

request = api.execute.getListOfUsersInGroup(idGroup = idOfGroup, offset = offset, v = 5.73, timeout = 120)

req()

b = request['users']

for i in b:

listOfMembers.append(str(i))

print(len(listOfMembers))

return (listOfMembers)

def getOperatorOfUser (phone):

global operators

action = False

if phone[0] == '+':

if phone[1] == '7':

action = True

if phone[0] == '7' or phone[0] == '8' or phone[0] == '9' or phone[0] == '(':

action = True

if action:

i = phone.find('9')

currOper = ''

if i <= 4:

currOper = phone[i:i + 3:1]

for oper in operators:

for code in operators[oper]:

if currOper == str(code):

return (oper)

return currOper

else:

return 'UnknownOper'

def getInfoOfUsers (listOfUsers):

localStrOfUsers = ''

totalCount = len(listOfUsers)

count = 0

localListOfUsers = []

for user in listOfUsers:

count += 1

localStrOfUsers = localStrOfUsers + user

if (count != 1000) and (totalCount - count > 0):

localStrOfUsers = localStrOfUsers + ','

else:

# print(localStrOfUsers)

tempListOfUsers = api.users.get(user\_ids = localStrOfUsers, fields = 'contacts', v = 5.83, timeout = 120)

req()

for user1 in tempListOfUsers:

localListOfUsers.append(copy.deepcopy(user1))

time.sleep(0.35)

totalCount -= count

count = 0

localStrOfUsers = ''

return (localListOfUsers)

def getGroupsOfUser (userID):

global officialGroups, isSetGroupOperator

localOffset = 0

result = api.execute.getGroups(idUser = userID, offset = localOffset, v = 5.74, timeout = 120)

req()

listOfGroups = []

for group in result['groups']:

for oGroup in officialGroups:

if str(group) == str(officialGroups[oGroup]):

listOfGroups.append(oGroup + ' ' + str(group) + ';')

while result['overCount'] > 0:

localOffset += 25000

result = api.execute.getGroups(idUser = userID, offset = localOffset, v = 5.74, timeout = 120)

req()

for group in result['groups']:

for oGroup in officialGroups:

if str(group) == str(officialGroups[oGroup]):

isSetGroupOperator[oGroup] = 1

listOfGroups.append(oGroup + ' ' + str(group) + ';')

return listOfGroups

def isContainOffGroup (listOfGroupsID):

global listOfOfficialGaroups

offGrousp = set(listOfOfficialGaroups)

userGroups = set(listOfGroupsID)

isContain = offGrousp & userGroups

if len(isContain) == 0:

return False

else:

return True

def getGroupsOfUsersTwo (listOfUsers):

global officialGroups, isSetGroupOperator, listOfOfficialGaroups

setOfOffGroups = set(listOfOfficialGaroups)

strOfIDs = ''

listOfUsersGroups = []

checkOffGroup = True

for i in range(len(listOfUsers)):

strOfIDs += str(listOfUsers[i])

if ((i + 1) % 25) and (i != (len(listOfUsers) - 1)):

strOfIDs += ','

if (not (i + 1) % 25) or (i == (len(listOfUsers) - 1)):

while True:

try:

request = api.execute.getGroupsTwo(strOfUsers = strOfIDs, v = 5.74, timeout = 120)

req()

break

except Exception as e:

print(e)

continue

strOfIDs = ''

result = request['list']

tmpListOfUsersGroups = list(filter(isContainOffGroup,list(result)))

for tmpList in tmpListOfUsersGroups:

tmpUserID = tmpList[0]

tmpList = list(setOfOffGroups & set(tmpList))

if checkOffGroup:

checkOffGroup = False

for offGroup in officialGroups:

if len(set([officialGroups[offGroup]]) & set(tmpList)) > 0:

isSetGroupOperator[offGroup] = 1

if isSetGroupOperator[offGroup] == 0:

checkOffGroup = True

if len(tmpList) != 0:

for j in range(len(tmpList)):

for offGroup in officialGroups:

if tmpList[j] == officialGroups[offGroup]:

tmpList[j] = offGroup

tmpList.insert(0,tmpUserID)

listOfUsersGroups.append(list(tmpList))

else:

listOfUsersGroups.append(list())

return listOfUsersGroups

def getWall (ID, offset):

try:

return api.execute.getWall(groupID = ID, offset = offset, v = 5.74, timeout = 120)

except Exception as e:

print(str(e))

print(';;;')

error = 'error'

return error

def getListOfPosts (ID, timeRange):

offset = 0

listOfPosts = []

boo = True

while boo:

wall = getWall(ID, offset)

req()

if wall != 'error':

for i in range(len(wall)):

if int(wall[i]['date']) < int(timeRange) and i != 0:

boo = False

break

else:

listOfPosts.append(copy.deepcopy(wall[i]))

offset += 2500

else:

print('error in getListOfPosts')

break

return listOfPosts

def getLikesOfPost (ID, listOfPosts):

listOfLikers = []

strOfIDs = ''

for i in range (len(listOfPosts)):

strOfIDs += str(listOfPosts[i]['id'])

if ((i + 1) % 25) and (i != (len(listOfPosts) - 1)):

strOfIDs += ','

if (not (i + 1) % 25) or (i == (len(listOfPosts) - 1)):

request = api.execute.getLikersOfPost(ownerID = ID, itemIDs = strOfIDs, v = 5.74, timeout = 120)

req()

strOfIDs = ''

for lst in request:

tempLst = []

for id in lst['items']:

tempLst.append(str(id))

listOfLikers.append(list(tempLst))

return listOfLikers

def getRepostsOfPost (ID, listOfPosts):

listOfLikers = []

strOfIDs = ''

for i in range(len(listOfPosts)):

strOfIDs += str(listOfPosts[i]['id'])

if ((i + 1) % 25) and (i != (len(listOfPosts) - 1)):

strOfIDs += ','

if (not (i + 1) % 25) or (i == (len(listOfPosts) - 1)):

request = api.execute.getRepostersOfPost(ownerID = ID, itemIDs = strOfIDs, v = 5.74, timeout = 120)

req()

strOfIDs = ''

for lst in request:

tempLst = []

for id in lst['items']:

tempLst.append(str(id))

listOfLikers.append(list(tempLst))

return listOfLikers

def countOfLikesAndReposts (userID, ListOfLikers, listOfReposters):

likes = 0

reposts = 0

for lst in ListOfLikers:

for id in lst:

if id == userID:

likes += 1

if likes > 0:

for lst in listOfReposters:

for id in lst:

if id == userID:

reposts += 1

response = {'likes' : likes, 'reposts' : reposts}

return response

def FindFriendsInGroups (userID, groupsOfFriends, friendsOfUser):

global globalIDForFindUsersList, officialGroups

saveThis = globalIDForFindUsersList

count = {'теле2' : 0, 'мтс' : 0, 'мегафон' : 0, 'билайн' : 0}

lstOfFriendsInGroup = []

for i in range(1, len(friendsOfUser)):

lstOfGroupsOfUser = []

globalIDForFindUsersList = friendsOfUser[i]

groupsOfFriend = list(filter(findUsersList, groupsOfFriends))

if len(groupsOfFriend) != 0:

groupsOfFriend = groupsOfFriend[0]

for j in range(1,len(groupsOfFriend)):

for offGroup in officialGroups:

if groupsOfFriend[j] == offGroup:

count[offGroup] += 1

lstOfGroupsOfUser.append(str(offGroup))

if len(lstOfGroupsOfUser) > 0:

lstOfFriendsInGroup.append({str(friendsOfUser[i]) : lstOfGroupsOfUser})

globalIDForFindUsersList = saveThis

return {'count' : count, 'lst' : lstOfFriendsInGroup}

def findUsersList(lst):

global globalIDForFindUsersList

if str(lst[0]) == str(globalIDForFindUsersList):

return True

else:

return False

def getFriendsOfUsers(listOfUsers):

strOfIDs = ''

globalFriends = []

listOfUsersFriends = []

for i in range(len(listOfUsers)):

strOfIDs += str(listOfUsers[i])

if ((i + 1) % 25) and (i != (len(listOfUsers) - 1)):

strOfIDs += ','

if (not (i + 1) % 25) or (i == (len(listOfUsers) - 1)):

result = api.execute.getFriends(strOfUsers = strOfIDs, v = 5.74, timeout = 120)

req()

strOfIDs = ''

for usersFriends in result:

listOfUsersFriends.append(list(usersFriends))

globalFriends += list(usersFriends)[1:]

globalFriends = list(set(globalFriends))

response = {'listOfFriends' : listOfUsersFriends, 'global' : globalFriends}

return response

def collectInfoFromUsers (groupID, timeRange):

global isSetGroupOperator, globalIDForFindUsersList, officialGroups

lstOfDataOfUsers = []

filteredLstOfDataOfUsers = []

WriteCSVFileOfUsers('users.csv', 'Empty', True)

WriteCSVFileOfUsers('usersProfitable.csv', 'Empty', True)

listOfMembersID = getListOfMembers(str(groupID))

print(datetime.datetime.now().strftime("%d-%m-%Y %H:%M"))

listOfUsers = getInfoOfUsers(listOfMembersID)

print(datetime.datetime.now().strftime("%d-%m-%Y %H:%M"))

listOdUsersGroups = getGroupsOfUsersTwo(listOfMembersID) #определение isSetGroupOperator происходит здесь

print(datetime.datetime.now().strftime("%d-%m-%Y %H:%M"))

print('Анализ друзей?')

startFriends = input()

if isSetGroupOperator['теле2']:

listOfPostsT2 = getListOfPosts(-18098621, timeRange)

listOfLikersT2 = getLikesOfPost(-18098621, listOfPostsT2)

listOfRepostersT2 = getRepostsOfPost(-18098621, listOfPostsT2)

if isSetGroupOperator['билайн']:

listOfPostsBee = getListOfPosts(-26514504, timeRange)

listOfLikersBee = getLikesOfPost(-26514504, listOfPostsBee)

listOfRepostersBee = getRepostsOfPost(-26514504, listOfPostsBee)

if isSetGroupOperator['мтс']:

listOfPostsMTS = getListOfPosts(-8458649, timeRange)

listOfLikersMTS = getLikesOfPost(-8458649, listOfPostsMTS)

listOfRepostersMTS = getRepostsOfPost(-8458649, listOfPostsMTS)

if isSetGroupOperator['мегафон']:

listOfPostsMega = getListOfPosts(-3785, timeRange)

listOfLikersMega = getLikesOfPost(-3785, listOfPostsMega)

listOfRepostersMega = getRepostsOfPost(-3785, listOfPostsMega)

for i in range(len(listOfUsers)):

DataOfUser = GetUser(listOfUsers[i])

for lst in listOdUsersGroups:

if lst:

if str(lst[0]) == str(DataOfUser['id']):

DataOfUser['groups'] = lst[1:len(lst):1]

break

if DataOfUser['groups']:

for oper in DataOfUser['groups']:

lstOfLikers = []

lstOfReposters = []

if oper == 'теле2':

lstOfLikers = listOfLikersT2

lstOfReposters = listOfRepostersT2

lOper = 'lTele2'

rOper = 'rTele2'

if oper == 'мтс':

lstOfLikers = listOfLikersMTS

lstOfReposters = listOfRepostersMTS

lOper = 'lMTS'

rOper = 'rMTS'

if oper == 'мегафон':

lstOfLikers = listOfLikersMega

lstOfReposters = listOfRepostersMega

lOper = 'lMegafon'

rOper = 'rMegafon'

if oper == 'билайн':

lstOfLikers = listOfLikersBee

lstOfReposters = listOfRepostersBee

lOper = 'lBeeline'

rOper = 'rBeeline'

response = countOfLikesAndReposts(DataOfUser['id'],lstOfLikers,lstOfReposters)

likes = response['likes']

reposts = response['reposts']

if likes:

DataOfUser[lOper] = likes

if reposts:

DataOfUser[rOper] = reposts

lstOfDataOfUsers.append(copy.deepcopy(DataOfUser))

if str(startFriends) == '1':

dictOfFriendsInGroup = {}

countOfUsersFriends = {}

lstOfOffGroups = []

filteredLstOfDataOfUsers = []

for group in officialGroups:

lstOfOffGroups.append(str(group))

setOfOffGroups = set(lstOfOffGroups)

lstOfAnalysisWithFriends = []

for DataOfUser in lstOfDataOfUsers:

setUserOper = set(list(DataOfUser['operator']))

setUserGroups = set(DataOfUser['groups'])

if (setOfOffGroups & setUserGroups) | (setOfOffGroups & setUserOper):

lstOfAnalysisWithFriends.append(DataOfUser['id'])

filteredLstOfDataOfUsers.append(copy.deepcopy(DataOfUser))

response = getFriendsOfUsers(lstOfAnalysisWithFriends)

globalFriends = response['global']

friendsOfUsers = response['listOfFriends']

for userFriends in friendsOfUsers:

countOfUsersFriends[userFriends[0]] = len(userFriends) - 1

with open('countOfUsersFriends.pickle', 'wb') as f:

pickle.dump(countOfUsersFriends, f)

with open('filteredLstOfDataOfUsers.pickle', 'wb') as f:

pickle.dump(filteredLstOfDataOfUsers, f)

groupsOfFriends = getGroupsOfUsersTwo(globalFriends)

for DataOfUser in filteredLstOfDataOfUsers:

if str(startFriends) == '1':

setUserOper = set(list(DataOfUser['operator']))

setUserGroups = set(DataOfUser['groups'])

if (setOfOffGroups & setUserGroups) | (setOfOffGroups & setUserOper):

saveThis = globalIDForFindUsersList

globalIDForFindUsersList = DataOfUser['id']

friendsOfUser = list(filter(findUsersList, friendsOfUsers))

if len(friendsOfUser) != 0:

response = FindFriendsInGroups(DataOfUser['id'], groupsOfFriends, list(filter(findUsersList, friendsOfUsers))[0])

localOper = response['count']

dictOfFriendsInGroup[DataOfUser['id']] = response['lst']

globalIDForFindUsersList = saveThis

DataOfUser['fTele2'] = localOper['теле2']

DataOfUser['fMTS'] = localOper['мтс']

DataOfUser['fMegafon'] = localOper['мегафон']

DataOfUser['fBeeline'] = localOper['билайн']

else:

DataOfUser['fTele2'] = 0

DataOfUser['fMTS'] = 0

DataOfUser['fMegafon'] = 0

DataOfUser['fBeeline'] = 0

else:

DataOfUser['fTele2'] = 0

DataOfUser['fMTS'] = 0

DataOfUser['fMegafon'] = 0

DataOfUser['fBeeline'] = 0

for DataOfUser in lstOfDataOfUsers:

WriteCSVFileOfUsers('users.csv', DataOfUser, False)

with open('dictOfFriendsInGroup.pickle', 'wb') as f:

pickle.dump(dictOfFriendsInGroup, f)

def analysisOfPageRank(dictOfFriendsInGroup,filteredLstOfDataOfUsers,countOfUsersFriends):

totalCountOfFriends = 0

lstOfFriendsInGroup = []

countOfFriendsFriends = {}

for user in countOfUsersFriends:

totalCountOfFriends += float(countOfUsersFriends[user])

for owner in dictOfFriendsInGroup:

for dict in dictOfFriendsInGroup[owner]:

for friend in dict:

lstOfFriendsInGroup.append(str(friend))

response = getFriendsOfUsers(lstOfFriendsInGroup)

lstOfFriendsFriends = response['listOfFriends']

for friends in lstOfFriendsFriends:

countOfFriendsFriends[str(friends[0])] = len(friends) - 1

totalCountOfFriends += len(friends) - 1

writeCSVFileOfAnalysisUsers('analysUsers.csv', 'Empty', True)

writeCSVFileOfAnalysisFriends('analysFriends.csv', 'Empty', True)

for DataUser in filteredLstOfDataOfUsers:

newDataUser = {'id':'', 'groups': '', 'friends': '', 'PageRank' : ''}

friendInfo = {'owner' : '', 'id':'', 'groups': '', 'friends': '', 'PageRank' : ''}

lstOfFriends = []

newDataUser['id'] = DataUser['id']

newDataUser['groups'] = DataUser['groups']

for count in countOfUsersFriends:

if str(DataUser['id']) == str(count):

newDataUser['PageRank'] = str(float(countOfUsersFriends[count])/totalCountOfFriends)

for owner in dictOfFriendsInGroup:

if str(owner) == str(DataUser['id']):

print()

print(dictOfFriendsInGroup[owner])

for dict in dictOfFriendsInGroup[owner]:

print(dict)

for friend in dict:

lstOfFriends.append(str(friend))

lstOfGroups = dict[friend]

friendInfo['owner'] = str(owner)

friendInfo['id'] = str(friend)

friendInfo['groups'] = lstOfGroups

for count in countOfFriendsFriends:

if str(count) == str(friend):

friendInfo['PageRank'] = str(float(countOfFriendsFriends[count])/totalCountOfFriends)

writeCSVFileOfAnalysisFriends('analysFriends.csv', friendInfo, False)

newDataUser['friends'] = lstOfFriends

writeCSVFileOfAnalysisUsers('analysUsers.csv', newDataUser, False)

def main(identifier):

print('Collect info from users: y/n')

while True:

print('Press the key')

startCollect = str(input())

if startCollect == 'y' or startCollect == 'Y':

collectInfoFromUsers(identifier, 1524245807)

break

else:

if startCollect == 'n' or startCollect == 'N':

print('aaa')

break

else:

print('Wrong key pressed')

with open('dictOfFriendsInGroup.pickle', 'rb') as f:

dictOfFriendsInGroup = pickle.load(f)

with open('filteredLstOfDataOfUsers.pickle', 'rb') as f:

lstOfDataOfUsers = pickle.load(f)

with open('countOfUsersFriends.pickle', 'rb') as f:

countOfUsersFriends = pickle.load(f)

analysisOfPageRank (dictOfFriendsInGroup,lstOfDataOfUsers,countOfUsersFriends)

main(53548055)