Научная статья

УДК 004.421.2:519.178
DOI 10.25205/1818-7900-2022-20-2-60-71

Модели импорта данных из мессенджера Telegram

Владимир Александрович Попов[1], Александр Андреевич Чеповский[2]

[1, 2]Национальный исследовательский университет «Высшая школа экономики»
Москва, Россия

[1]vapopov_1@edu.hse.ru
[2]aachepovsky@hse.ru

Аннотация

В данной работе описан алгоритм импорта данных из мессенджера Telegram и построения взвешенных графов взаимодействующих объектов. Для импорта данных за основу берутся заданные Telegram-каналы. Далее итерационно выявляются каналы, имевшие любое из зафиксированных трех взаимодействий с предыдущими: общие внешние ссылки, упоминания друг друга, репосты. Далее алгоритм ориентируется на заданную конфигурацию и по ней вычисляет веса на ребрах полученного графа. Конфигурация учитывает тип взаимодействия каналов между собой. Авторы вводят понятие (U, M, R)-модели информационного взаимодействия. Авторы описывают разработанный алгоритм и реализованное программное обеспечение для построения взвешенных графов. В статье приведен пример взвешенного графа взаимодействующих объектов, построенного описанным алгоритмом по (U, M, R)-модели.

Ключевые слова

анализ социальных сетей, импорт данных из социальных сетей, модели информационного взаимодействия, выделение сообществ

Для цитирования

Попов В. А., Чеповский А. А. Модели импорта данных из мессенджера Telegram // Вестник НГУ. Серия: Информационные технологии. 2022. Т. 20, № 2. С. 60–71. DOI 10.25205/1818-7900-2022-20-2-60-71

Telegram Messenger Data Import Models

Vladimir A. Popov[1], Alexander A. Chepovskiy[2]

[1, 2]National Research University Higher School of Economics
Moscow, Russian Federation

[1]vapopov_1@edu.hse.ru
[2]aachepovsky@hse.ru

Abstract

In this paper, an algorithm to import data from the messenger Telegram and to build weighted graphs of interacting objects is described. To import data, the given Telegram-channels are taken as a basis. Then, channels which had any of the recorded three interactions with previous ones are iteractively revealed: common external links, mentions of each other, reposts. Further, the algorithm focuses on the given configuration and uses it to calculate the weights on the edges of the resulting graph. The configuration takes into account the type of interaction of channels with each other. The authors introduce the concept of (U, M, R)-model of information interaction. The authors describe the developed algorithm and implemented software for constructing weighted graphs. The paper shows the example of weighted graph of interacting objects that was built by the described algorithm according to the (U, M, R)-model.

Keywords

social network analysis, import of data from social networks, information interaction models, community detection

For citation

Popov V. A., Chepovskiy A. A. Telegram Messenger Data Import Models. Vestnik NSU. Series: In-formation Technologies, 2022, vol. 20, no. 2, p. 60–71. (in Russ.) DOI 10.25205/1818-7900-2022-20-2-60-71

Введение

Тема изучения графов взаимодействующих объектов, в том числе полученных из социальных сетей, давно и активно изучается [1–4]. Но в настоящее время в мире крайне интенсивно растет популярность не только социальных сетей, но и кроссплатформенной системы обмена сообщениями Telegram. Она используется не только для обмена личными сообщениями между пользователями, но и все активнее как площадка для профессиональных каналов – новых медиа, издающих регулярные посты-публикации разной направленности. За счет удобной возможности цитирования других каналов осуществляется быстрое распространение информации. Все это позволяет рассматривать множество Telegram-каналов как граф взаимодействующих объектов, что приводит к задаче выделения в этой сети неявных сообществ [5–8].

В данной работе представлен алгоритм импорта данных из мессенджера Telegram, позволяющий построить взвешенный граф взаимодействующих объектов для дальнейшего анализа как его структуры, так и текстовой информации, распространяемой каналами. Вначале приведено подробное описание разработанной авторами *(U, M, R)*-модели информационного взаимодействия в мессенджере Telegram по аналогии с *(F, L, C, R)*-моделью для Twitter [9].

Мессенджер Telegram

Telegram – мессенджер, в котором помимо личных и групповых чатов пользователи могут создавать свои Telegram-каналы, представляющие собой блоги. Есть как открытые каналы, которые могут читать все пользователи мессенджера, так и закрытые, доступ к которым предоставляется по приглашению или прямой ссылке. У каждого канала есть уникальный ID, уникальное имя (ChannelName), ссылка и краткое описание. Каждый пользователь Telegram также имеет свой уникальный ID и может указать свое уникальное имя (UserName). У каждого пользователя есть возможность подписаться на любой открытый канал, таким образом, у канала формируется список читателей, но данный список виден только администраторам канала, обычным пользователям доступна информация только о количестве читателей канала.

Сам канал представляет собой ленту записей авторов. Каждая запись может содержать текст, медиа файлы (картинки, видео- и аудиозаписи), ссылки на внешние сайты или другие Telegram-каналы, упоминания других каналов. Все записи содержат отметку времени, порядковый ID номер, информацию о количестве просмотров другими пользователями этой записи и другую информацию. Также сама запись может быть репостом с другого Telegram-канала (у автора канала есть возможность просто сделать репост с другого канала или репост с указанием своего комментария) или ответом на какой-либо пост. Из дополнительных возможностей: у авторов канала есть функционал для организации комментариев других пользователей под каждым постом и возможность редактирования уже опубликованных постов.

Таким образом, учитывая функционал ведения каналов в мессенджере Telegram, в данной работе нас будут интересовать следующие поля записей в канале: уникальный ID, время, текст поста, список внешних ссылок, список упоминаний других Telegram-каналов и отметка является ли пост репостом (если является, то также интересует название канала, откуда совершен репост). А для идентификации каналов мы будем использовать их уникальное имя (ChannelName) и ID.

На основе этого функционала было выделено 3 вида взаимоотношений (взаимодействий) между каналами в мессенджере Telegram:

– наличие в постах общих внешних ссылок (URL) у двух разных каналов;
– упоминание одним каналом другого канала в тексте поста;
– репост одним каналом сообщения другого канала.

Учитывая описанные взаимосвязи между каналами мессенджера Telegram, формируется граф взаимодействующих объектов, где сами каналы являются вершинами, а ребра строятся

на основании имевших место взаимодействий. Причем, так как эти взаимодействия не равноценны, целесообразно рассматривать взвешенные графы. Вес на ребрах будет посчитан как линейная комбинация весов, соответствующих каждому из зафиксированных взаимоотношений каналов. Эти отдельные веса для взаимодействий были приняты как параметры модели, описанной ниже.

(U, M, R)-модель информационного воздействия

Определим *(U, M, R)*-модель информационного воздействия в мессенджере Telegram как взвешенный граф *G(V, E)*, у которого *V* – множество вершин (каналов), *E* – множество ребер – взаимодействий между парами каналов. На данном множестве ребер *E* определим весовую функцию $w(e_{AB})$ $(e_{AB} \in E;\ A, B \epsilon V)$ (следующим образом:

$$w(e_{AB}) = U \times \delta^{U}_{e_{AB}} + M \times \delta^{M}_{e_{AB}} + R \times \delta^{R}_{e_{AB}},$$ где

$\delta^{U}_{e_{AB}}$ – количество общих уникальных внешних ссылок (URL) в постах у каналов *A* и *B*;
$\delta^{M}_{e_{AB}}$ – количество постов, где в тексте канал *A* упомянул канал *B* плюс количество постов, где *B* упомянул *A*;
$\delta^{R}_{e_{AB}}$ – количество репостов каналом *A* сообщений канала *B* плюс количество репостов канала *A* у канала *B*.

Рассмотрим на примере расчет коэффициентов и итогового веса ребра.

Пусть канал *A* в своих постах ссылается два раза на *url_1*, один раз на *url_2* и четыре раза на *url_3*, а канал *B* один раз на *url_1*, два раза на *url_3* и по четыре раза на сайты *url_4*, *url_5*. Тогда общие уникальные внешние ссылки у каналов *A* и *B* – это *url_1* и *url_3*, соответственно количество 2, коэффициент $\delta^{U}_{e_{AB}} = 2$.

Пусть канал *A* в своих постах 2 раза упоминает канал *B*, а канал *B* в своих постах 3 раза упоминает канал *A*. Тогда коэффициент $\delta^{M}_{e_{AB}} = 2 + 3 = 5$.

Аналогично, пусть канал *A* в своей ленте 2 раза репостнул посты канала *B*, а канал *B* 1 раз репостнул *A*. Тогда коэффициент $\delta^{R}_{e_{AB}} = 2 + 1 = 3$.

Получаем, что итоговая формула для подсчета веса ребра e_{AB} будет следующей:
$w(e_{AB}) = U \times 2 + M \times 5 + R \times 3$, где коэффициенты *U*, *M*, *R* задаются пользователем.
При значениях *(U, M, R)* = (1, 2, 3), получаем $w(e_{AB}) = 1 \times 2 + 2 \times 5 + 3 \times 3 = 21$.

Описанный механизм вычисления весов ребер применяется для каждой пары Telegram-каналов из числа вершин графа. И на основе этих значений формируется итоговый взвешенный граф взаимодействующих объектов.

Другие подходы к построению графов взаимодействующих объектов мессенджера Telegram

Идея построить граф взаимодействующих объектов на основе данных из Telegram-каналов не нова. С ростом популярности мессенджера предлагались различные подходы для решения этой задачи.

Один из подходов к построению графов сети Telegram авторы описывают в [10]. При данном подходе информация о взаимодействиях и связях между каналами берется из описания этих каналов. Авторы рассматривают все Telegram-каналы, на которые подписан начальный пользователь (или можно просто взять определенный изначальный список каналов), далее для каждого канала смотрят его описание и в этих описаниях ищут упоминание других пользователей или каналов. И соответственно между каналом и указанными каналами/пользователями в описании строится связь.

Например, ниже приведено описание Telegram-канала *@Reddit*:

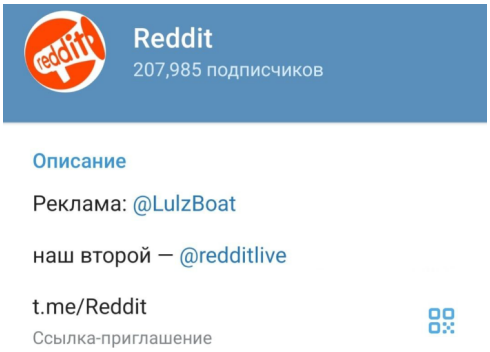


Рис. 1. Описание Telegram-канала @Reddir
Fig. 1. Description of the Telegram channel @Reddit

Как можно увидеть, в описании присутствует упоминание канала *@redditlive* и пользователя *@LulzBoat*. Соответственно, между этими объектами и каналом *@Reddit* строится ребро в итоговом графе взаимодействующих объектов. В данном случае вершинами графа являются не только каналы, но и обычные пользователи (или боты) мессенджера Telegram.

Для определенных задач такой подход может быть полезен. Например, если нужно найти каналы, которые имеют одного рекламного менеджера. Тут мы видим возможность по доработке этого подхода. Например, если анализировать описание каналов полностью и выделять какие-то ключевые слова и соотносить каналы по тематике на основе их описаний. Но при таком подходе можно выделить только явные связи между объектами в сети Telegram и выделить только явные сообщества, что не всегда дает полной информации о сети. Также из минусов можно выделить то, что не у всех каналов есть подробное описание, и не везде авторы каналов упоминают какие-либо дополнительные контакты, поэтому далеко не все связи в графе будут получены. Поэтому, наряду с простотой реализации, такой подход получается менее информативным, нежели предложенная в нашей работе *(U, M, R)*-модель информационного воздействия.

Другой подход к построению графов взаимодействующих объектов – это модель, при которой вершинами графа являются Telegram-каналы, а ребра формируются на основе взаимосвязей между ними (наличие упоминаний и репостов). В [11] описывается три модели: Обычный граф, Обычный взвешенный граф и Двудольный взвешенный граф. Первая модель предполагает, что между двумя вершинами-каналами есть ребро, если хотя бы в одном посте одного канала был упомянут другой канал или один канал репостнул другой. Но при таком подходе теряется информация о силе связей между узлами сети, поэтому авторы вводят вторую модель – Обычный взвешенный граф. При данной модели помимо наличия связей между двумя каналами считается и количество связей, причем упоминания и репосты вносят вклад в вес ребер в равной мере, таким образом предложенная модель является аналогом *(U, M, R)*-модели информационного воздействия с параметрами $U=0, M=R=1$. Соответственно модель, предложенная в [11], является частным случаем модели, описанной в предыдущем разделе, а поэтому является и менее гибкой для дальнейшего использования. Помимо этих двух моделей в [11] описан и другой подход к построению графов, а именно – модель двудольного взвешенного графа. Каналы разделяются на две категории (каналы могут относиться к двум категориям одновременно): каналы-авторы (Authors) и каналы-переадресаторы (Forwarders). Если канал A репостнул или упомянул канал B, то канал A относится к каналам-переадресаторам, а канал B к каналам-авторам. Соответственно первые каналы являются источниками оригинальной информации, а вторые – каналами, которые распространяют информацию каналов-авторов. И в данном подходе итоговая модель информационного воздействия представляет собой двудольный граф, где вершинами являются Telegram-каналы, а ребра строятся аналогично преды-

дущему подходу. Отличие состоит в том, что если один канал упоминает или репостит другие каналы, а его, в свою очередь, упоминают или репостят другие каналы, то данный канал входит в граф два раза: как автор и как переадресатор. Данный подход имеет определенный плюс: помимо близости между вершинами он позволяет установить направление связей и роль узлов в сети. Но данный подход также можно доработать, используя *(U, M, R)*-модель с общими параметрами.

Также стоит отметить, что в статье [11] авторы берут готовые данные с ресурса [12] и рассматривают только те посты, в которых есть упоминание других каналов или которые являются репостами. Соответственно не рассматриваются тексты каналов, поэтому не могут быть выделены общие внешние ссылки или источники между каналами, а также не могут быть скачаны тексты каналов для их дальнейшего анализа.

Реализация импорта данных из мессенджера Telegram

Для формирования описанных выше взвешенных графов взаимодействующих объектов было разработано приложение, включающее в себя функционал как импорта данных из мессенджера Telegram, так и построения графов на основе скачанных данных. На рис. 2 показан основной интерфейс разработанной программы. В верхней части экрана представлены функции для импорта данных, а в нижней части – для построения графов:

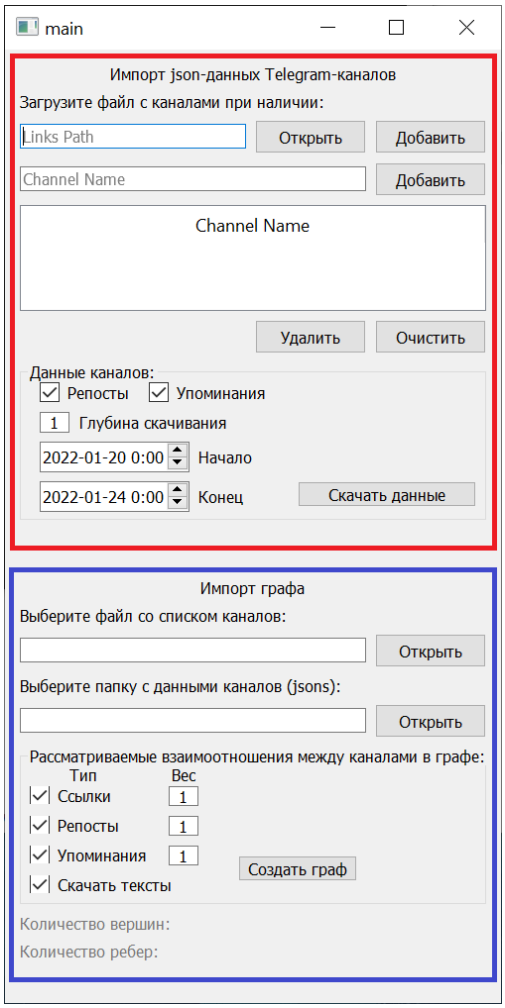


Рис. 2. Основной интерфейс с двумя разделами
Fig. 2. The main interface with two sections

Основным способом для получения данных из Telegram является использование API (Application Programming Interface) [13]. Telegram предоставляет разработчикам обширный интерфейс для работы с мессенджером: можно сделать бота или свой кастомизированный клиент. С помощью различных функций можно реализовать проверку новых сообщений или записей в каналах, автоответ на сообщения, пересылки и т. д. А также можно обрабатывать данные с каналов: получать информацию о каналах, скачивать список записей с их полными атрибутами, скачивать комментарии к записям.

Для использования Telegram API прежде всего необходимо получить *api_id* и *api_hash*, для этого на сайте Telegram в разделе для разработчиков (URL: https://my.telegram.org/auth) нужно создать приложение с кратким описанием. После получения необходимых полей, с помощью различных инструментов можно работать с API.

В данной работе использовалась библиотека Python Telethon [14], которая представляет собой инструмент для взаимодействия с Telegram API в качестве пользователя (создавая объект клиента Telegram API) или бота.

Общая логика построения графа взаимодействующих объектов

Общая постановка задачи подразумевает, что на вход алгоритмам будет подаваться список Telegram-каналов, промежуток времени, глубина скачивания графа и список типов взаимодействия, по которым будет расширяться граф на новую глубину. Последнее подразумевает один из трех вариантов: упоминания/репосты, либо оба этих типа взаимодействия. При импорте данных и построении связей будут рассматриваться посты в каналах только за указанный промежуток времени. Значение глубины скачивания учитывается по следующей логике:

– при глубине 0 вершинами итогового графа будут только исходно заданные каналы. Обозначим этот список каналов как ChannelList_0;

– при глубине 1 рассматриваются исходные каналы и каналы, связанные с ними (в постах исходных каналов ищутся репосты и/или упоминания других каналов в указанном промежутке времени). Обозначим объединенный список как ChannelList_1;

– на шаге k берутся каналы из списка ChannelList_(k-1) и аналогично в зависимости от исходного выбора ищутся упоминания и/или репосты, что в совокупности дает список новых каналов, которые связаны с ChannelList_(k-1). Объединяя, получаем список ChannelList_k.

Все эти параметры задаются в первом блоке приложения (рис. 3).

Например, в указанном случае будет скачана информация за декабрь 2021 года от трех каналов (@sportsru, @sportexpress, @FootballTheSun) на глубину 3, расширяясь за счет репостов и упоминаний. Исходно заданные каналы могут как выражать единую позицию, так и наоборот, транслировать противоположные мнения. Также в исходном списке может быть только один

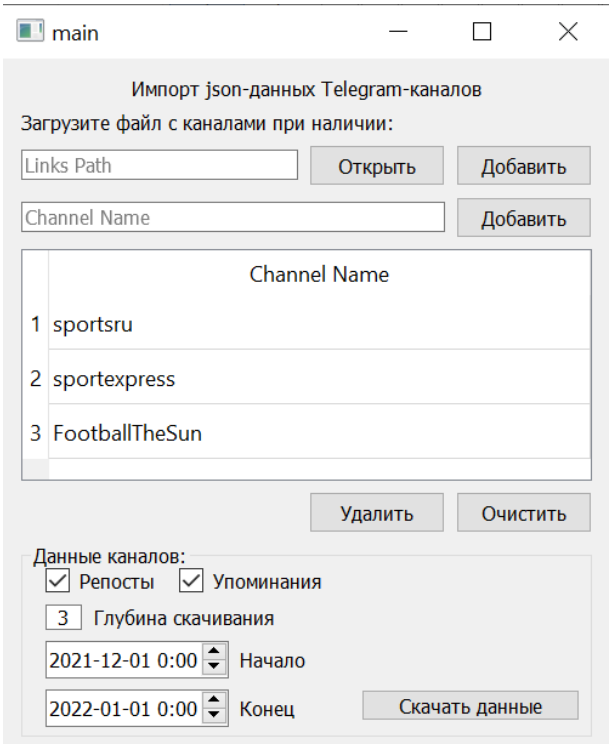


Рис. 3. Первый блок приложения для импорта вершин
Fig. 3. The first block of the vertex import application

канал, интересный оператору для анализа. Никаких особых ограничений на специфику и количество каналов нет.

Далее в тексте рассмотрим детально, как получить списки каналов с репостами, упоминаниями и списки внешних ресурсов, на которые были приведены ссылки в постах. Этот основной шаг используется итерационно при построении множества вершин графа взаимодействующих объектов. После формирования множества вершин на следующем шаге строятся ребра взвешенного графа в соответствии с конкретной выбранной *(U, M, R)*-моделью.

Импорт данных при помощи Telegram API

Рассмотрим решение следующей задачи: задано название или ID Telegram-канала и временной интервал, в пределах которого смотрим посты, необходимо получить следующие списки:

1) список каналов, которые производили репосты заданного канала, с количеством репостов по каждому каналу;

2) список упоминаний других Telegram-каналов, с количеством упоминаний по каждому каналу (считаются упоминания в разных постах, два упоминания одного канала в одном посте считается за одно упоминание);

3) список уникальных внешних ресурсов/сайтов, на которые были ссылки в постах.

Если задан ID канала, то для начала получаем специальный объект *entity*, который в дальнейшем используется для вызова функций API. Делается это с помощью функции *get_entity(ChannelID)* библиотеки Python Telethon, где аргумент *ChannelID* – уникальный *ID* канала. Таким образом, *ID* канала сопоставляется специализированный объект *Channel*. В этом объекте хранится множество данных о канале, одним из которых является значение *ChannelName*, в дальнейшем будем его использовать при построении связей между каналами. Если изначально задано *ChannelName* канала, то предварительная обработка не нужна. Стоит отметить, что функция *get_entity(ChannelID)* является тяжелой функцией, поэтому количество запросов API данного типа существенно ограничено.

Затем для канала импортируются данные по всем постам канала за заданный промежуток времени. Для этого используется функция *GetHistoryRequest()*, которая в нашем случае получает на вход *ChannelName* и более позднюю дату (дату, с которой в обратном хронологическом порядке будут скачиваться посты). Учитывая особенности API, скачивание происходит по 300 постов за каждый запуск функции до тех пор, пока не импортируется более ранний пост относительно заданного временного интервала или не закончатся посты в канале. Также из-за ограничений API на частоту запросов, между запусками указанной функции делаются искусственные паузы. И в итоге, мы имеем скорость импорта примерно 100 записей канала в секунду.

Далее полученные сырые данные обрабатываются, форматируются и сохраняются в json формате. В результате по всем постам скачивается полная информация: ID поста в канале, дата, дата редактирования (если есть), текст, медиа объекты, является ли данный пост ответом или репостом (если является, то также указывается ID канала, с которого совершен репост), информация о наличии гиперссылок (как и на внешние сайты, так и ссылки на каналы Telegram) и т. д.

Следующий этап заключается в получении из этих данных необходимых трех списков. Данные по списку репостов получить проще всего: проходим по всем постам и смотрим на поле *fwd_from*, если оно не равно None, то в поле *from_id* хранится ID канала, с которого был произведен репост. А информацию по другим двум спискам извлечь немного сложнее, необходимо парсить исходный текст постов и вспомогательную информацию о наличии гиперссылок [15–16].

В результате описанных выше действий, для указанного канала формируются три объекта: множество репостов (ID Telegram-каналов), множество упоминаний (ChannelName Telegram-каналов) и множество уникальных ссылок на внешние ресурсы. Данный алгоритм в дальнейшем используется в виде функции.

Формирование множества вершин графа

Напомним, что в самом начале оператор задает список исходных каналов, глубину скачивания графа, рассматриваемые типы взаимоотношений и промежуток времени.

Если глубина графа задана нулевой, то итоговый список вершин графа взаимодействующих объектов – это только изначально заданные каналы. Поэтому для каждой вершины исходного списка запускается процедура, указанная ранее, и формируется json-данные с каждого канала, которые впоследствии будут использованы для построения ребер графа.

При глубине графа равной $N > 0$, будет N раз запущена итерация скачивания данных. На каждой итерации для получения списка вершин графа глубины K берем список вершин для глубины $K-1$. Сначала для каждого Telegram-канала из этого списка скачиваем два списка: список репостов и список упоминаний каждого канала, потом удаляем исходные каналы. Таким образом, получаем список каналов, которые расширяют граф на следующую глубину. Далее для каждого нового канала запускаем импорт данных.

Необходимо отметить, что в зависимости от изначально выбранных типов взаимодействий впоследствии в графе могут возникнуть изолированные вершины. Это происходит, например, в случае *(U, 0, 0)*-модели. Формирование вершин идет до построения ребер, поэтому возможна ситуация, когда вершины были добавлены в силу упоминаний и/или репостов каналов. Но при этом отсутствие ссылок на общие внешние ресурсы приведет к нулевому весу ребер. Поэтому в финальном графе возможно наличие изолированных вершин.

Реализация алгоритма для построения графов

Как неоднократно было сказано ранее, мы выделяем 3 типа взаимоотношений между каналами в мессенджере Telegram: наличие в постах общих внешних ссылок у двух разных каналов, упоминание одним каналом другого канала в текстах постов, репост одним каналом сообщений другого канала. И каждое из этих взаимодействий вносит свой вклад в итоговый вес ребра между вершинами графа взаимодействующих объектов.

На вход алгоритму подается список каналов-вершин графа, папка, где хранятся json-файлы, скачанные для этих каналов в рамках импорта данных с Telegram, а также подаются коэффициенты *(U, M, R)*-модели, по которым будет посчитан итоговый вес ребер. Стоит сказать, что если оператор не хочет рассматривать какую-либо связь между каналами при формировании ребер, то он просто выбирает соответствующий коэффициент равный нулю. Данные настройки происходят во втором блоке приложения (рис. 4).

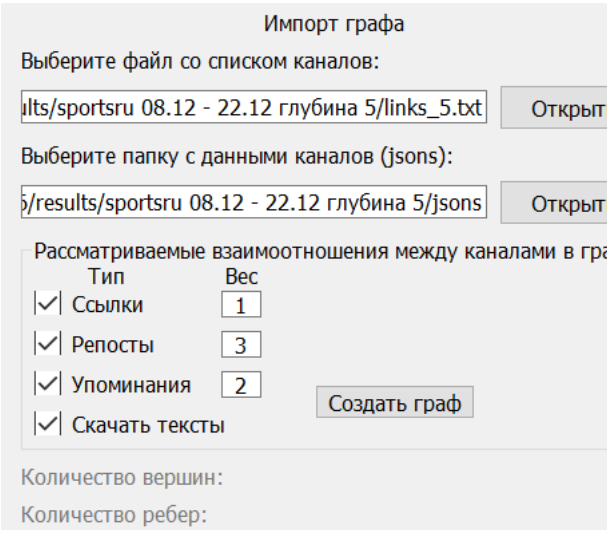


Рис. 4. Второй блок приложения для построения ребер
Fig. 4. The second block of the application for constructing edges

Итоговый вес ребра e_{AB} между каналами A и B будет посчитан как взвешенная сумма:

$$w(e_{AB}) = U \times \delta_{e_{AB}}^{U} + M \times \delta_{e_{AB}}^{M} + R \times \delta_{e_{AB}}^{R}$$

Далее для каждого канала из указанного файла смотрится json-файл и формируется список каналов, которые были упомянуты в текстах канала, список репостов данного канала и список внешних ссылок. После этого для каждой пары вершин происходит подсчет веса ребра:

1) Если параметр *U* выбран больше 0, то для каждой пары вершин смотрятся списки уникальных внешних ресурсов/сайтов, на которые были ссылки в постах. И эти два списка пересекаются между собой. Количество полученных элементов есть коэффициент $\delta_{e_{AB}}^{U}$.

2) Если параметр *M* выбран больше 0, то для каждой пары вершин смотрятся словари упоминаний. И складываются значения: сколько раз первый канал в текстах своих постов упомянул второй канал и сколько раз второй канал упомянул первый. Полученное значение равняется коэффициенту $\delta_{e_{AB}}^{M}$.

3) Если параметр *R* выбран больше 0, то для каждой пары вершин смотрятся словари репостов. И складываются значения: сколько раз первый канал сделал репост сообщений второго канала и сколько раз второй канал репостнул первый. Полученное значение – коэффициент $\delta_{e_{AB}}^{R}$.

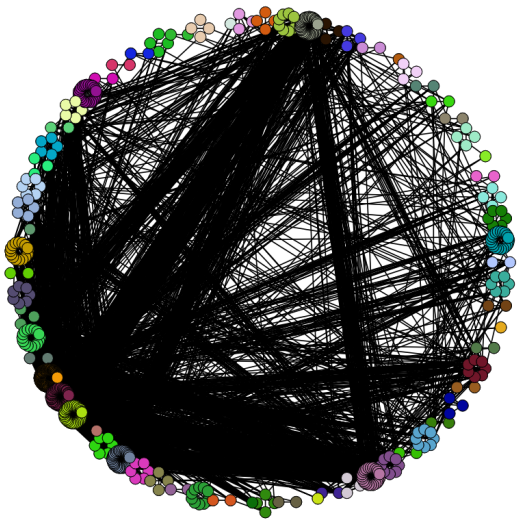
Таким образом, для каждой пары вершин A и B вычисляются коэффициенты и по формуле считается вес ребра $w(e_{AB})$. Соответственно при $w(e_{AB}) = 0$ ребра между вершинами нет, а при большем 0, формируется ребро с посчитанным весом.

И по итогу строится финальный взвешенный граф взаимодействующих объектов, который впоследствии сохраняется в специальный XML-подобный формат AVS (специальный формат для хранения графов, в котором описываются атрибуты каждой вершины и каждого ребра) для дальнейшей обработки и анализа. В нашем случае атрибутом ребер является рассчитанный по формуле вес, а атрибутами вершин – названия каналов и ссылки на файлы, в которых хранятся все тексты данного канала за заданный промежуток времени.

Пример

В качестве примера 22.12.2021 был произведен импорт данных за две недели (8.12-22.12) от стартового источника, которым был выбран спортивный Telegram-канал @sportsru, на глубину 5. При построении графа взаимодействующих объектов использовалась *(U, M, R)*-модель для значений (1, 2, 3). Таким образом, вес ребер определялся по следующей формуле:

$$w(e_{AB}) = 1 \times \delta_{e_{AB}}^{U} + 2 \times \delta_{e_{AB}}^{M} + 3 \times \delta_{e_{AB}}^{R}$$



Больше всего вес добавляют в такой версии репосты, меньше всего – общие упоминания внешних ресурсов. Полученный граф содержал 590 вершин (каналов Telegram) и 4352 ребра со средним весом 7,93. На данном графе с помощью алгоритма Louvain [17] было выделено 69 неявных сообществ (рис. 5). Такой импорт данных и построение взвешенного графа взаимодействующих объектов позволяют в дальнейшем провести изучение структуры и текстов подсети Telegram-каналов.

Рис. 5. Пример графа для канала @sportsru
Fig. 5. Example graph for the @sportsru channel

Заключение

В рамках данной работы авторами предложена *(U, M, R)*-модель информационного взаимодействия каналов Telegram. Для нее построен описан алгоритм импорта данных из мессенджера и построения взвешенного графа взаимодействующих объектов. Такая структура полезна для дальнейшего комплексного анализа импортированной подсети, текстов каналов и их направленности. Авторам видится актуальным дальнейшее изучение подходов по анализу графов и текстов, полученных с использованием *(U, M, R)*-модели.

Список литературы

1. **Лещёв Д. А., Сучков Д. В., Хайкова С. П., Чеповский А. А.** Алгоритмы выделения групп общения // Вопросы кибербезопасности. 2019. Т. 32. № 4. С. 61–71.
2. **Соколова Т. В., Чеповский А. А.** Анализ профилей сообществ социальных сетей // Системы высокой доступности. 2018. Т. 14, № 3. С. 82–86.
3. **Коломейченко М. И., Поляков И.В., Чеповский А.А., Чеповский А. М.** Выделение сообществ в графе взаимодействующих объектов // Фундаментальная и прикладная математика. 2016. Т. 21. № 3. С. 131–139.
4. **Roth M., Ben-David A., Deutscher D.** Suggesting Friends Using the Implicit Social Graph – KDD'10, July 25–28, 2010, Washington, DC, USA. 2010.
5. **Girvan M., Newman M.** Community structure in social and biological networks. Proceedings of the National Academy of Sciences. 2002. Vol. 99. No 12. P. 7821–7826.
6. **Blondel V. D., Guillaume J. L., Lambiotte R., Lefebvre E.** Fast unfolding of communities in large networks // Journal of Statistical Mechanics: Theory and Experiment. 2008. No 10. P10008. – 12 p.
7. **Rosvall M.** The map equation / M. Rosvall, D. Axelsson, C. T. Bergstrom // The European Physical Journal Special Topics. – 2009.
8. **Chepovskiy A. A., Leshchev D. A., Khaykova S. P.** Core Method for Community Detection, in: Complex Networks & Their Applications IX. Volume 1: Proceedings of the Ninth International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2020. Springer, 2021. P. 38–50. DOI 10.1007/978-3-030-65347-7_4.
9. **Попов В. А., Чеповский А. А.** Модели импорта данных из Твиттера // Вестник НГУ. Серия: Информационные технологии. 2021. Т. 19, № 2. С. 76–91. DOI 10.25205/1818-7900-2021-19-2-76-91.
10. Building graph for Telegram chats, channels and their neighbors. URL: https://ntwrk.today/2020/04/09/building-telegram-graph.html (дата обращения: 11.03.2022).
11. **Tikhomirova K., Makarov I.** Community Detection Based on the Nodes Role in a Network: The Telegram Platform Case, in: 9th International Conference, AIST 2020, Skolkovo, Moscow, Russia, October 15–16, 2020, Revised Selected Papers.
12. TGStat. URL: https://tgstat.ru/ (дата обращения: 11.03.2022).
13. Telegram API. URL: https://core.telegram.org/api (Дата обращения: 16.02.2022).
14. Библиотека Python Telethon. URL: https://docs.telethon.dev/en/stable/ (дата обращения: 16.02.2022).
15. **Mitchell R.** Web Scraping with Python. Sebastopol: O'Reilly Media, 2015
16. Библиотека Python Beautiful Soup. URL: https://www.crummy.com/software/BeautifulSoup/ (дата обращения: 16.02.2022).
17. **Que X., Checconi F., Petrini F., Gunnels J.** Scalable Community Detection with the Louvain Algorithm // 29th IEEE International Parallel & Distributed Processing Symposium, May 25–29, 2015.

References

18. **Leschyov D. A., Suchkov D. V., Khaykova S. P., Chepovskiy A. A.** Algorithms to reveal communication groups // Voprosy kiberbezopasnosti. 2019. 32(4). P. 61–71. (in Russ.). DOI 10.21681/2311-3456-2019-4-61-71.
19. **Sokolova T. V., Chepovskij A. A.** Analiz profilej soobshhestv social`ny`x setej // Sistemy` vy`sokoj dostupnosti. 2018. T. 14, № 3. P. 82–86. (in Russ.)
20. **Kolomejchenko M. I., PolyakovI. V., Chepovskij A. A., Chepovskij A. M.** Vy`delenie soobshhestv v grafe vzaimodejstvuyushhix ob``ektov // Fundamental`naya i prikladnaya matematika. 2016. Vol. 21. №3. P. 131–139. (in Russ.)
21. **Roth M., Ben-David A., Deutscher D.** Suggesting Friends Using the Implicit Social Graph – KDD'10, July 25–28, 2010, Washington, DC, USA., 2010.
22. **Girvan M., Newman M.** Community structure in social and biological networks. Proceedings of the National Academy of Sciences. 2002. Vol. 99. No 12. P. 7821–7826.
23. **Blondel V. D., Guillaume J. L., Lambiotte R., Lefebvre E.** Fast unfolding of communities in large networks // Journal of Statistical Mechanics: Theory and Experiment. 2008. No 10. P10008. 12 p.
24. **Rosvall. M.** The map equation / M. Rosvall, D. Axelsson, C. T. Bergstrom // The European Physical Journal Special Topics. – 2009.
25. **Chepovskiy A. A., Leshchev D. A., Khaykova S. P.** Core Method for Community Detection, in: Complex Networks & Their Applications IX. Volume 1: Proceedings of the Ninth International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2020. Springer, 2021. P. 38–50. DOI 10.1007/978-3-030-65347-7_4.
26. **Popov V. A., Chepovskiy A. A.** Twitter Data Import Models // Vestnik NSU. Series: Information Technologies. 2021. Vol. 19. No. 2. P. 76–91. (in Russ.). DOI 10.25205/1818-7900-2021-19-2-76-91.
27. Building graph for Telegram chats, channels and their neighbors. URL: https://ntwrk.today/2020/04/09/building-telegram-graph.html.
28. **Tikhomirova K., Makarov I.** Community Detection Based on the Nodes Role in a Network: The Telegram Platform Case, in: 9th International Conference, AIST 2020, Skolkovo, Moscow, Russia, October 15–16, 2020, Revised Selected Papers.
29. TGStat. URL: https://tgstat.ru/.
30. Telegram API. URL: https://core.telegram.org/api.
31. Python Telethon. URL: https://docs.telethon.dev/en/stable/.
32. **Mitchell R.** Web Scraping with Python. Sebastopol: O'Reilly Media, 2015.
33. Python Beautiful Soup. URL: https://www.crummy.com/software/BeautifulSoup/.
34. **Que X., Checconi F., Petrini F., Gunnels J.** Scalable Community Detection with the Louvain Algorithm // 29th IEEE International Parallel & Distributed Processing Symposium, May 25–29, 2015.

Сведения об авторах

Попов Владимир Александрович, студент магистратуры, Национальный исследовательский университет «Высшая школа экономики» (Москва, Россия).

Чеповский Александр Андреевич, кандидат физико-математических наук, доцент, Национальный исследовательский университет «Высшая школа экономики» (Москва, Россия).

Information about the Authors

Popov A. Vladimir, master's student, National Research University Higher School of Economics (Moscow, Russian Federation)

Alexander A. Chepovskiy – Ph.D. (mathematics), Associate Professor, National Research University Higher School of Economics (Moscow, Russian Federation)

Статья поступила в редакцию 26.04.2022;
одобрена после рецензирования 12.05.2022; принята к публикации 12.05.2022
The article was submitted 26.04.2022;
approved after reviewing 12.05.2022; accepted for publication 12.05.2022