УДК 004.42

Е.В. Никитина, С.А. Карпунин, Д.Р. Карпенко

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

РАЗРАБОТКА ИНСТРУМЕНТА ДЛЯ ПОИСКА СВЯЗЕЙ МЕЖДУ ПОЛЬЗОВАТЕЛЯМИ «ВКОНТАКТЕ»

Социальные сети играют большую роль в жизни современного общества. Огромное количество людей сохраняют в них личные данные, что делает социальные сети источником ценной информации, такой как личные и профессиональные связи и интересы, принадлежность к различным сообществам. Данная статья описывает эффективный способ решения проблемы поиска связей между пользователями социальной сети. В качестве источника данных использовалась социальная сеть "ВКонтакте". В результате исследования было разработано приложение, решающее поставленную задачу за приемлемое время. В основе алгоритма лежит двунаправленный поиск в ширину, а для хранения данных используется графовая СУБД.

Социальные сети, графы, поиск в графе

Введение

Поиск социальных связей между людьми в настоящее время может быть применён в различных сферах, таких как реализация почтовых спам-фильтров[1], поиск возможных друзей в социальных сетях, таргетированная реклама.

С ростом популярности социальных сетей стало значительно легче получать данные о личных и профессиональных связях между людьми, однако, несмотря на открытость информации о подобных отношениях, в открытом доступе практически нет доступных решений для их обработки и анализа. Ручной поиск является неэффективным и требует многократного повторения однотипных действий, что поднимает проблему автоматизации поискового процесса.

Целью данной работы является разработка приложения для поиска дружественных связей между пользователями социальной сети "ВКонтакте".

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Определить эффективный алгоритм поиска кратчайшего пути
2. Выбрать наиболее эффективный способ представления и хранения данных о пользователях и их дружеских связях
3. Разработать приложение для поиска связей между пользователями сети "ВКонтакте"

Обзор подходов к решению проблемы

С точки зрения анализа данных социальная сеть может быть представлена в виде графа, вершины которого соответствуют пользователям сети, а ребра - отношению дружбы.

Таким образом, задача определения связи между двумя заранее заданными людьми переходит в задачу поиска пути в невзвешенном графе с высоким коэффициентом ветвления (branching factor)[2]. Большое количество вершин и ребер исключает использование алгоритмов, требующих представление в памяти всего графа целиком (Dijkstra, Floyd–Warshall)[3].

Динамичность структуры графа в совокупности с его размером дает основание считать данный граф бесконечным, что, в свою очередь, также ограничивает выбор алгоритмов поиска. Так, в случае бесконечного графа, поиск в глубину может заблудиться в одной из его частей, не содержащих искомой вершины, без возможности вернуться обратно. Поиск в ширину, напротив, гарантирует нахождение пути к вершине, если она существует[4]. Однако, поиск в глубину имеет преимущество по памяти.

Подходящим алгоритмом поиска при вышеперечисленных условиях является двунаправленный поиск, так как имеет наименьшую сложность и гарантию завершения. Данный алгоритм эффективен в графах с

малым диаметром и высоким коэффициентом ветвления и в простейшей реализации представляет собой запуск двух поисков в ширину, начинающихся из исходной и конечной вершин соответственно.

Алгоритм двунаправленного поиска использовался при решении проблемы поиска связей для социальной сети “Twitter”[5]. В данной работе исследуются различные алгоритмы поиска с точки зрения скорости работы и минимизации запросов к API социальной сети. В результате исследования было установлено, что двунаправленный поиск в ширину с ограничением глубины является наиболее эффективным по обоим параметрам.

Существующие решения проблемы поиска связей для социальной сети “ВКонтакте” обладают рядом недостатков. Так, приложение [6] основывается на последовательном построении деревьев друзей для двух пользователей и дальнейшем извлечении цепочек в случае существования пересечения данных деревьев. Программа выполняется непосредственно в браузере и для цепочек длиной более 4 расход времени и памяти становится критично большим. Также недостатком данного решения является неудачный выбор инструментов для визуализации графов, что снижает возможность интерпретации результатов работы программы.

На основании обзора подходов к решению проблемы можно сделать вывод о том, что наиболее эффективным алгоритмом как по времени, так и по расходу памяти является двунаправленный поиск в ширину с ограничением глубины.

Выбор метода решения

На основании обзора аналогов можно сформулировать следующие требования к решению поставленной проблемы.

Решение должно обладать простым интерфейсом, предоставляющим возможность выбора пользователей сети “ВКонтакте” для поиска связей между ними. Для наглядного отображения результатов работы приложения полученную информацию о дружеских связях следует показывать в виде графа.

Подсчет и отображение затраченного на поиск времени, а также других количественных характеристик выполнения алгоритма (длина кратчайшего пути между заданными вершинами, количество сохраненных пользователей), даст возможность анализировать работу программы и сравнивать ее с другими существующими решениями.

Критически важным качеством является приемлемая скорость работы программы. В условиях ограничений публичного API “ВКонтакте” и работы с большими объемами данных приложение должно решать задачу за минимальное время. В [5] были приведено исследование времени поиска. В среднем, для цепочек длины 4 время поиска составило 1.4 минуты. Это значение можно использовать в качестве основы для оценки эффективности - время работы при схожих условиях не должно его превышать. Для выполнения этого требования алгоритм работы программы должен быть основан на двунаправленном поиске в ширину. Кроме того, необходима корректная обработка вырожденных случаев, в которых, например пользователи уже являются друзьями или кто-то из них, напротив, не имеет друзей.

Согласно [7], в данных условиях реляционные базы данных значительно уступают графовым как по памяти, так и по производительности, поэтому целесообразно в качестве способа представления данных выбрать именно графовую СУБД.

Описание решения

Разработанное приложение представляет собой веб-сервис, состоящий из клиентской и серверной части. Пользователю предоставляется простой интерфейс для ввода id двух пользователей и поиска связи между ними. В процессе работы отображается текущее состояние приложение. Результирующие данные представлены в виде интерактивного графа. Кроме того, приложение отображает статистику о времени работы и объёме проанализированных данных.

Технологии

Интерфейс приложения реализован с помощью JavaScript библиотеки React.js. Для поддержки контейнера состояния подключена библиотека Redux. Вывод полученного графа и диаграммы статистики осуществляется с использованием библиотек JGraph и Charts.js соответственно.

Все перечисленные инструменты являются открытым программным обеспечением и распространяются по лицензии MIT, разрешающей безвозмездное использование.

Серверная часть приложения написана с использованием фреймворка Spring MVC. Клиент-серверное взаимодействие реализовано с помощью REST API. Аутентификация и авторизация пользователя осуществляется по протоколу OAuth2.0 через учетную запись сети “ВКонтакте”.

В качестве инструмента для хранения информации о пользователях используется графовая база данных Neo4j, так как она имеет встроенное API для поиска кратчайших путей и эффективного импорта большого объема данных.

Архитектура программной реализации

Модель данных

Анализируемые данные представляются в виде ненаправленного невзвешенного графа, вершины которого содержат идентификаторы пользователей “ВКонтакте”, ребра реализуют отношение "дружба" типа многие-ко-многим.

В целях разделения на кластеры множества узлов, с которыми работают пользователи, вершины имеют дополнительный атрибут, содержащий идентификатор пользователя, инициализирующего работу поискового алгоритма.

Для обхода ограничений Community версии Neo4j, связанных с невозможностью создания составных ключей, вершины графа содержат результат конкатенации идентификаторов вершины и кластера. Уникальность полученной строки позволяет использовать её в качестве ключа.

Так как для каждого пользователя создается отдельный кластер, общий объём хранимых пользователей может в разы превышать количество пользователей "ВКонтакте". В связи с быстрым устареванием анализируемых данных, постоянное хранение полученных графов ради оптимизации поиска нецелесообразно. После завершения работы алгоритма, соответствующий кластер вершин удаляется из базы данных, что позволяет существенно экономить дисковое пространство.

Алгоритм

Входными данными разработанного алгоритма являются идентификаторы двух пользователей сети “ВКонтакте”, выходными - все найденные кратчайшие пути между вершинами, соответствующими заданным пользователям. Каждая итерация алгоритма состоит из нескольких стадий:

1. Получение данных в JSON формате через публичное API “Вконтакте”. Для каждого из необработанных пользователей, находящихся в очереди, запрашивается список его друзей. В связи с ограничениями, установленными API “ВКонтакте” на объем запрашиваемой информации, вся очередь разделяется на партии размером 25 пользователей, для каждой из которых выполняется отдельный запрос.

2. Трансформация полученных данных в CSV формат. API “ВКонтакте” возвращает запрашиваемые данные в JSON формате, неудобном для импорта в базу данных. Для повышения скорости импорта осуществляется трансформация полученной информации в файлы в .csv формате.

3. Импорт .csv файлов в базу данных. Neo4j предоставляет инструмент LOAD CSV, позволяющий одним запросом импортировать большой объем данных в .csv формате. Синтаксис команды LOAD CSV подразумевает указание местоположения импортируемых файлов в виде URI или пути в файловой системе. Для уменьшения связности компонентов приложения предоставляется отдельный веб-интерфейс для получения .csv файлов.

4. Проверка наличия пути между указанными пользователями. Выполняются запросы, используя готовые методы Neo4j API для поиска кратчайших путей между заданными вершинами.

Если в результате очередной итерации путь обнаружен, из базы данных экспортируется соответствующий подграф и возвращается пользователю.

Согласно теории шести рукопожатий[8], диаметр исследуемого графа не превышает шести. Данная теория многократно проверялась на практике[9],[10]. Опираясь на результаты данных исследований, можно утверждать, что ограничение общего количества итераций алгоритма до трех с высокой вероятностью по-прежнему будет находить путь между двумя вершинами.

Таким образом, общая схема алгоритма работы программы выглядит следующим образом:

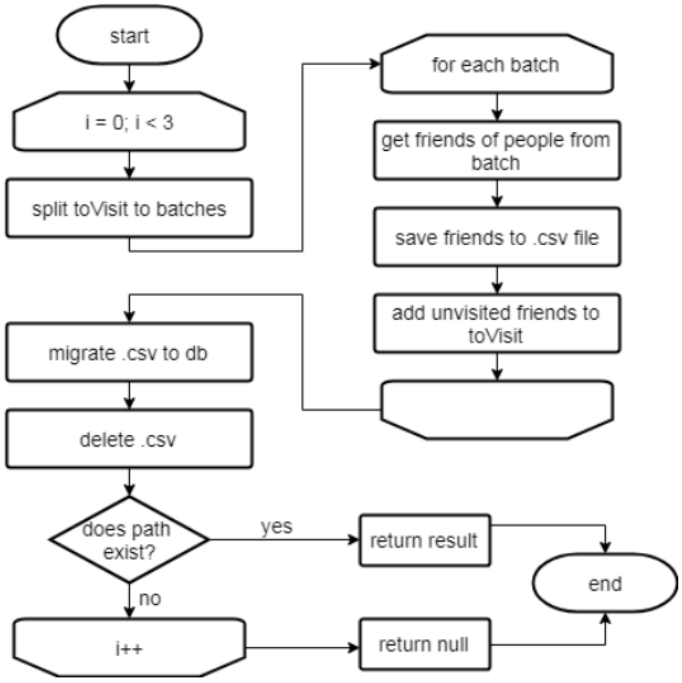

Рис.1

Структура реализации

Разработанный продукт представляет собой клиент-серверное приложение с многослойной архитектурой (multilayered architectural pattern). Общая схема приложения изображена на рис.2.

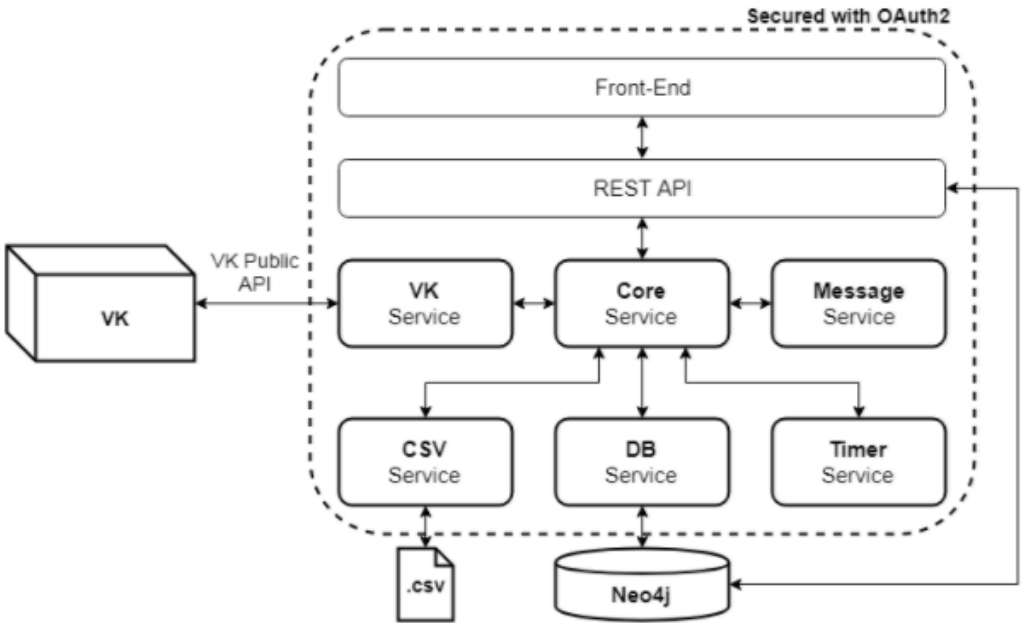

Рис.2

UI слой. Структурно интерфейс состоит из формы ввода двух идентификаторов пользователей сети “ВКонтакте”, между которыми нужно найти связь (рис. 3), и панели с результатами работы программы.

Панель, в свою очередь, содержит следующую информацию:

- граф кратчайших путей (рис. 3)
- время, затраченное на каждую из стадий работы алгоритма (рис. 4)
- другие статистические данные: длина пути, количество пользователей, сохраненных в базу данных

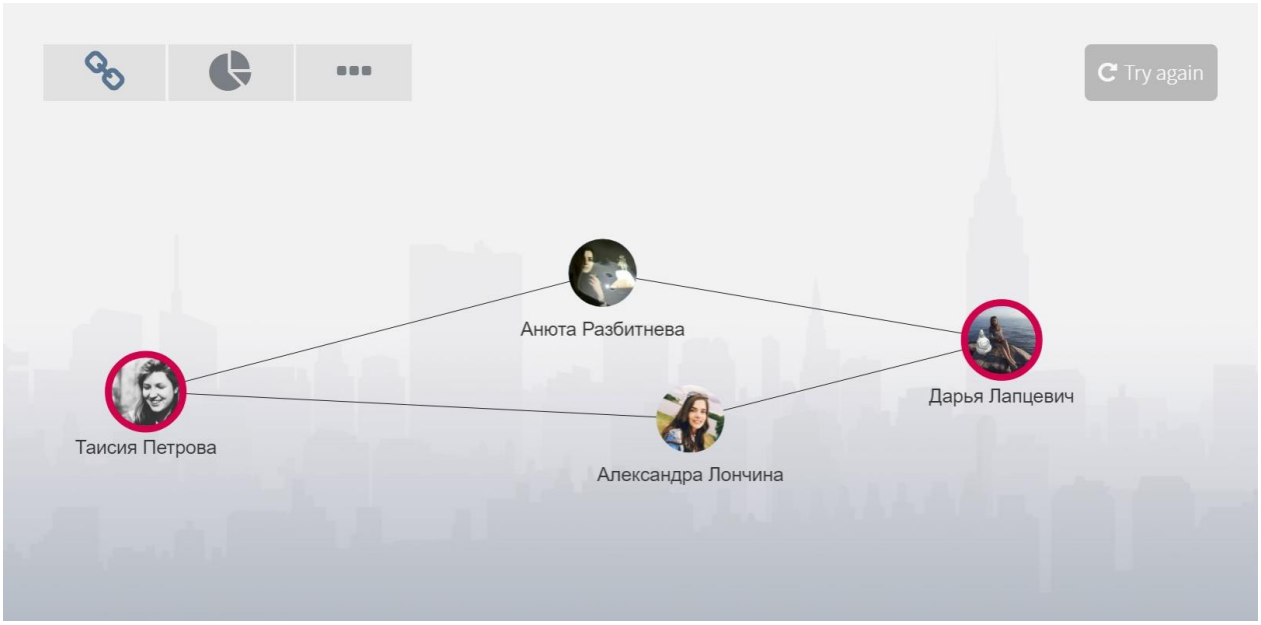

Рис.3

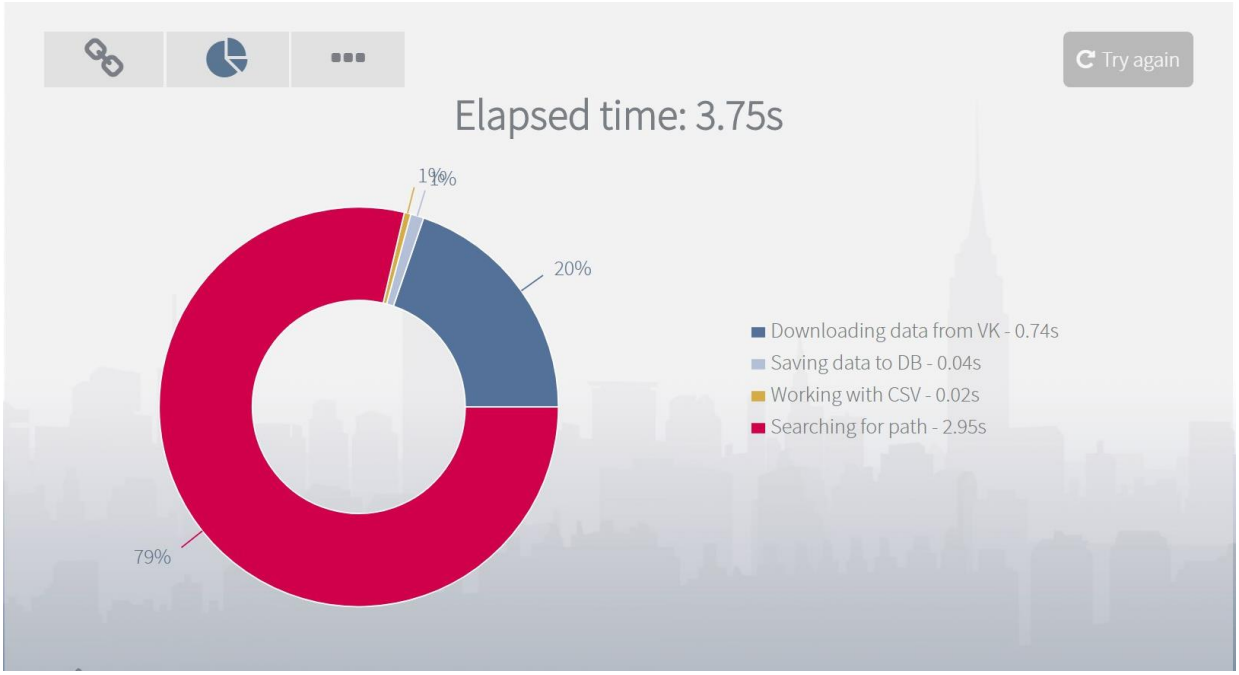

Рис.4

Service слой. Сервисный слой приложения состоит из шести компонентов, отвечающих за различный стадии работы программы.

Core service является ядром приложения. В его обязанности входит поддержка алгоритма двунаправленного поиска и своевременный вызов других компонентов сервисного слоя.

Message service отвечает за отправление сообщений на веб-интерфейс приложения посредством WebSocket канала.

Timer service используется для подсчета времени, затрачиваемого на каждую из стадий работы программы.

DB service осуществляет взаимодействие с сервером базы данных Neo4j, предоставляя интерфейс для посылки запросов на языке Cypher.

VK service обеспечивает аутентификацию и авторизацию пользователя через протокол OAuth2.0 по гранту Authorization Code Grant. Также в обязанности этого компонента входит получение данных о пользователях социальной сети, таких как имя, фамилия, фотография и список друзей.

Данный сервис инкапсулирует работу с методами JavaSDK VK библиотеки для использования основных функций публичного API сети “ВКонтакте”.

Для улучшения производительности работы с API используется метод execute, позволяющий поместить до 25 обращений к методам API в одном запросе. Для поддержки такого способа получения информации были написаны фрагменты кода в VKScript-формате.

CSV service отвечает за хранение и удаление временных файлов в .csv формате.

Persistence слой. Информация, необходимая для поиска кратчайшего пути в графе, хранится на сервере базы данных Neo4j и в файловой системе в виде временных файлов в формате .csv.

Заключение

На основании обзора существующих подходов к решению поставленной проблемы и анализа их достоинств и недостатков был предложен способ нахождения дружественных связей между пользователями сети “ВКонтакте”. Описанный алгоритм является достаточно эффективным по времени выполнения, количеству запросов к API “ВКонтакте” и расходуемой памяти.

Эффективность работы программы по времени обусловлена лежащим в ее основе алгоритмом двунаправленного поиска в ширину с ограничением глубины. Эффективность по расходуемой памяти достигается за счет использования графовой базы данных и отказа от длительного хранения кластеров вершин.

Разработанное приложение[11],[12] решает проблему поиска дружественных связей в социальное сети “ВКонтакте”. Слабая связность компонентов приложения предполагает возможность кастомизации приложение (другие источники и механизмы хранения данных). Дальнейшая работа будет направлена на добавление критериев поиска связей, таких как город, место работы или учёбы. Так же будет проведено исследование скорости работы в зависимости от различных характеристик, таких как длина пути и объём сохранённых данных.

СПИСОК ЛИТЕРАТУРЫ

1. Пат. US 7945674 B2 Degrees of separation for handling communications / Barry Appelman Опубл. 17 май 2011.
2. Свободная энциклопедия // Wikipedia. URL: https://en.wikipedia.org/wiki/Branching_factor (дата обращения 07.12.2017)
3. Р. Седжвик, К. Уэйн Фундаментальные алгоритмы на Java: Издательский дом «Вильямс», 2013. 575-603с.
4. Coppin, B. (2004). Artificial intelligence illuminated. Jones & Bartlett Learning. 79–80 с.
5. Degrees of Separation in Social Networks / R. Bakhshandeh, M. Samadi, Z. Azimifar, J. Schaeffer // Proceedings, The Fourth International Symposium on Combinatorial Search // Castell de Cardona, Barcelona, Spain, Июль 15.16, 2011
6. Whom I know // Github URL: http://artfultom.github.io/whom-i-know/dist/index.html (дата обращения 07.12.2017)
7. Benchmarking database systems for social network applications / R. Angles, A. Prat-Perez, D. Dominguez-Sal, J. LLariba-Pey // Proceeding GRADES '13 First International Workshop on Graph Data Management Experiences and Systems Article No. 15 New York, New York — Июль 23 - 23, 2013
8. S. Milgram The Small World Problem // Psychology Today 1967 том. 2, 60-67 с
9. J. Travers, S. Milgram An Experimental Study of the Small World Problem // Sociometry 1969 том 32, №4, 425-443 с
10. The Anatomy of the Facebook Social Graph / J. Ugander, B. Karrer, L. Backstrom, C. Marlow // arXiv 18.11.2011
11. Репозиторий на github с исходным кодом // Github URL: https://github.com/moevm/nosql-2017-six_handshakes
12. Образ на docker hub // hub.docker.com URL: https://hub.docker.com/r/evnikitina/6shakes/

E.V. Nikitina, S.A. Karpunin, D.R. Karpenko

Saint Petersburg Electrotechnical University “LETI”

Development of the instrument for searching for relations between the users of “VKontakte”

Social networks play an important role in the life of modern society. A huge number of people retain personal data in networks, and that makes social networks a source of valuable information, such as personal and professional connections and interests, participation in different communities. This article describes the effective way to solve the problem of finding connections between social network users. The social network "VKontakte" was used as a data source. As a result of the study, was developed an application that solves the task at an acceptable time. The algorithm is based on bidirectional wide-width search, and a graph DBMS is used to store data.