Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №6
по курсу
”Проектирование Рекомендательных Систем”

Тема Алгоритмы выявления сообществ

Студент Якуба Д. В.

Группа ИУ7-33М

Оценка (баллы) _____

Преподаватели Быстрицкая А.Ю.

Москва — 2023 г.

Оглавление

ВВЕДЕНИЕ

Цель работы – исследовать алгоритмы выявления сообществ.

Для достижения поставленной цели потребуется:

- привести описание алгоритма Label Propogation Communities;
- привести описание алгоритма Лювина;
- привести описание алгоритма Fluid Communities;
- привести описание используемых для исследования данных;
- привести визуализацию результатов;
- провести анализ скорости работы алгоритмов.

1. Аналитический раздел

1.1 Алгоритмы выявления сообществ

Алгоритмы выяления сообществ – это методы анализа сетей, направленные на выделение групп узлов в сети, которые тесно взаимодействуют друг с другом. Сообщества в сетях обычно определяются на основе структуры графа взаимосвязей между узлами.

1.2 Label Propogation Communities

LPC или алгоритм распространения меток основан на распространении информации через сеть. Этот алгоритм часто используется для выявления сообществ в графах, где узлы схожи между собой.

Основные шаги алгоритма:

1. Инициализация меток – каждый узел в графе ининциализируется с некоторой начальной меткой, это может быть уникальный идентификатор или любая другая информация;
2. Распространение меток – метки начинают распространяться по сети, на каждом шаге узлы обновляют свои метки на основе меток соседей; Чаще всего узел принимает метку, которая преобладает среди его соседей;
3. Итерирование – шаги распространения повторяются до тех пор, пока метки не стабилизируются или до достижения максимального числа итераций;
4. Финализация – после завершения итераций, каждый узел устанавливает свою финальную метку, которая была определена в результате многократного распространения и обновления.

Алгоритм не всегда сходится к однозначному результату, а финальный метки могут сильно зависеть от начальной конфигурации.

1.3 Алгоритм Лювина

Данный алгоритм основан на оптимизации меры модулярности, которая измеряет качество разбиения графа на сообщества.

Основные шаги алгоритма:

1. Инициализация – с самого начала каждый узел считается отдельным сообществом;
2. Оптимизация модулярности – алгоритм производит попытки объеди-

нять сообщества с целью максимизации модулярности графа (модулярность – мера качества разбиения графа на сообщества, оценивает, насколько хорошо внутригрупповые связи сильнее, чем случайные связи между узлами);

3. Шаги объединения и переразбиения – алгоритм проходит через граф несколько раз, выполняя два основных шага: объединение и переразбиение; На шаге объединения узлы объединяются в еще более крупные сообщества, а на шаге переразбиения происходит оптимизация внутригрупповых связей путем перемещения узлом между сообществами;
4. Оптимизация модулярности – после каждого объединения или переразбиения происходит оптимизация модулярности, чтобы убедиться, что изменение соответствует улучшению структуры графа;

4.1 Шаги объединения и переразбиения производятся до тех пор, пока модулярность не перестанет увеличиваться либо же по достижению заданного числа итераций.

Алгоритм Лювена известен своей способностью быстро выявлять сообщества в крупных сетях.

1.4 Fluid Communities

Данный алгоритм основан на идее введения ряда флюидов (то есть сообществ) в неоднородную среду (то есть в графе, не являющимся полным графом), где флюиды будут расширяться и воздействовать друг на друга под влиянием топологии среды до достижения устойчивого состояния.

К достоинствам алгоритма относят его асинхронность, возможность задать количество обнаруживаемых сообществ путем определения начального количества флюидов, а также избегание создания больших сообществ.

2. Конструкторский раздел

2.1 Источник данных

В качестве источника данных был взят датасет, располагающийся в свободном доступе на веб-сайте Snap Stanford [1]. Набор данных содержит информацию о связях пользователей в социальной сети Facebook (Meta – запрещенная на территории РФ организация), всего в ней 4039 узлов и 88234 ребер.

3. Технологический раздел

3.1 Средства реализации

В качестве используемого был выбран язык программирования Python [2].

Данный выбор обусловлен следующими факторами:

- Большое количество исчерпывающей документации;
- Широкий выбор доступных библиотек для разработки;
- Простота синтаксиса языка и высокая скорость разработки.

При написании программного продукта использовалась среда разработки Visual Studio Code. Данный выбор обусловлен тем, что данная среда распространяется по свободной лицензии, поставляется для конечного пользователя с открытым исходным кодом, а также имеет большое число расширений, ускоряющих разработку.

3.2 Библиотеки

При анализе и обработке датасета, а также для решения поставленных задач использовались библиотеки:

- pandas;
- numpy;
- matplotlib;
- networkx [3].

Данные библиотеки позволили полностью покрыть спектр потребностей при выполнении работы.

4. Исследовательский раздел

4.1 Условия исследований

Исследование проводилось на персональном вычислительной машине со следующими характеристиками:

- процессор Apple M1 Pro,
- операционная система Ventura 13.5.2,
- 32 Гб оперативной памяти.

4.2 Визуализация графа

На рисунке 4.1 представлена визуализация графа без использования средств построения макета.

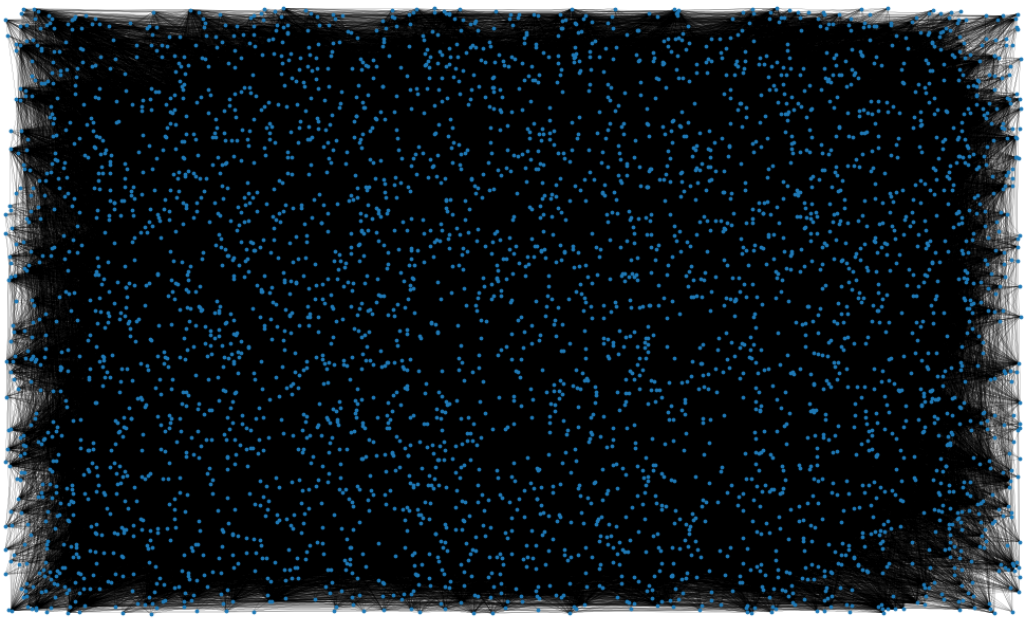

Рис. 4.1: Визуализация графа.

На рисунках 4.2 – 4.4 представлена визуализация графа с использованием функции построения макета **spring layout** при количестве итераций 15, 30 и 50 соответственно.

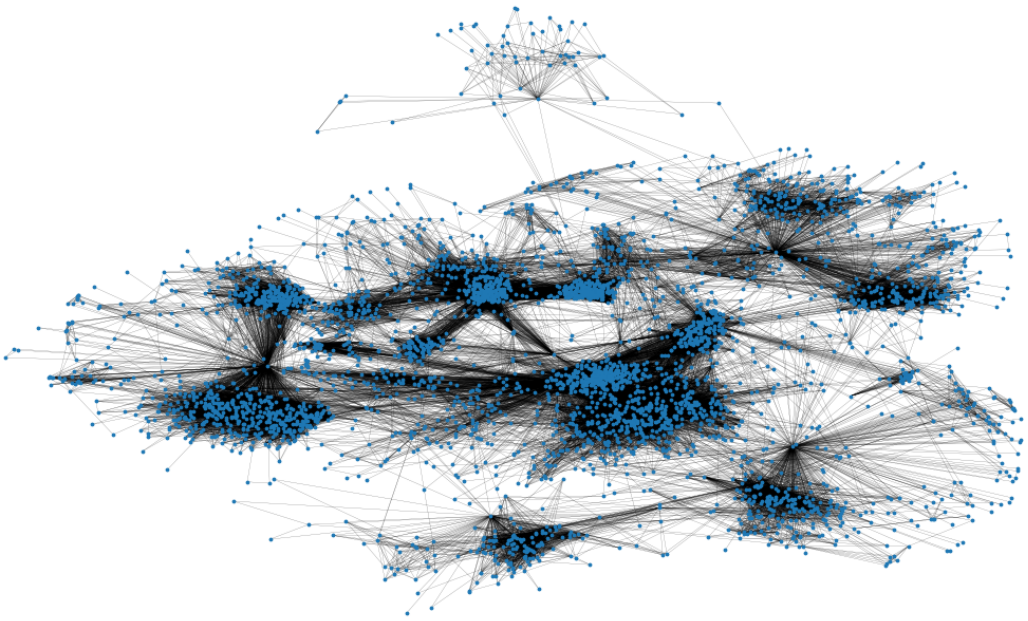

Рис. 4.2: Визуализация графа с использованием функции **spring layout**, количество итераций 15.

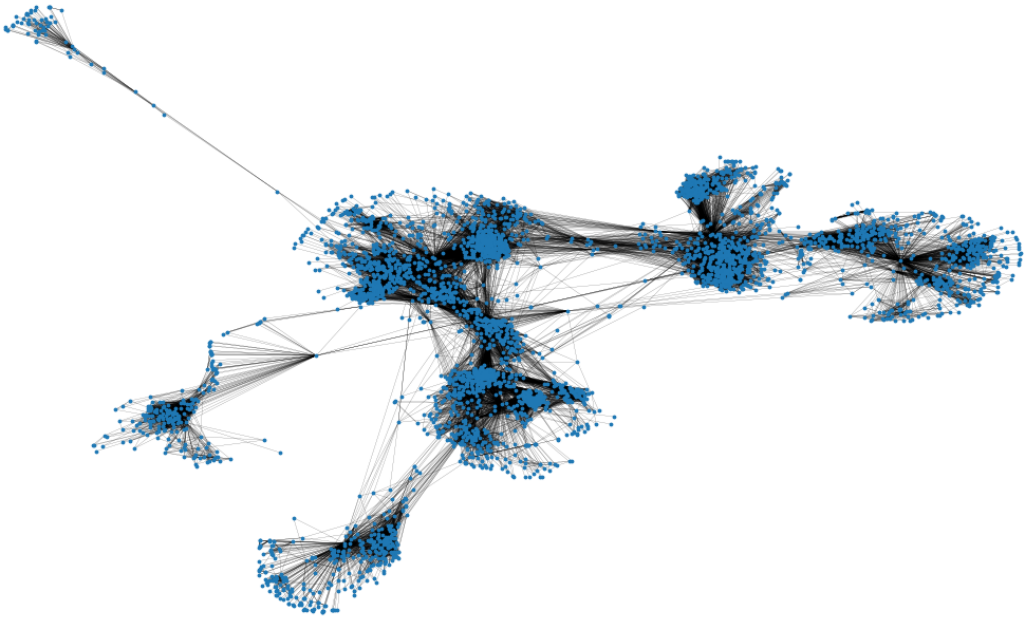

Рис. 4.3: Визуализация графа с использованием функции **spring layout**, количество итераций 30.

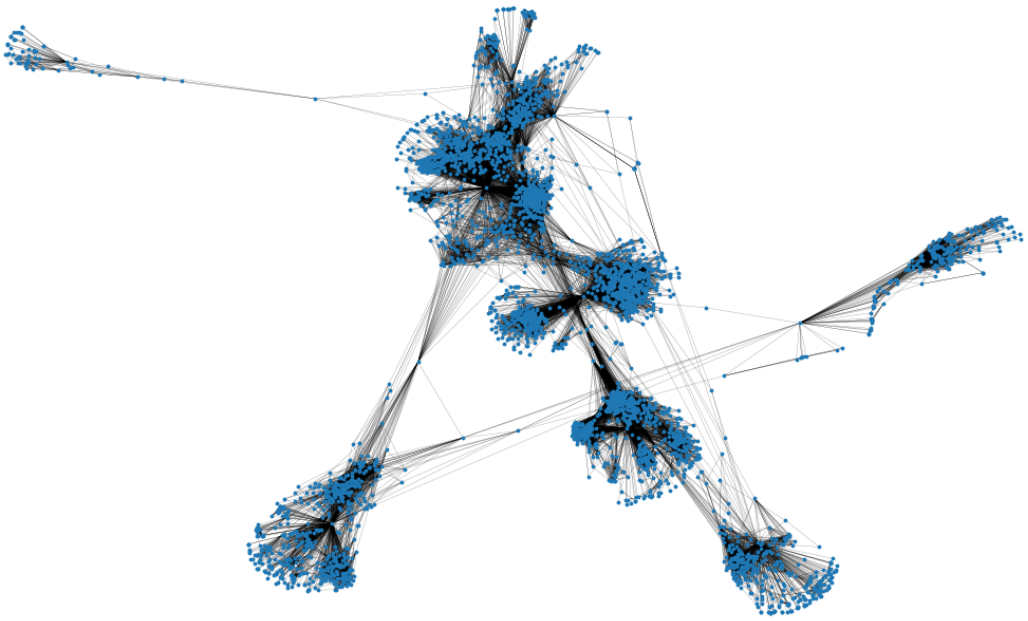

Рис. 4.4: Визуализация графа с использованием функции **spring layout**, количество итераций 50.

В качестве используемого при визуализации результатов работы алгоритмов обнаружения сообществ будет использоваться граф, визуализированный при количестве итераций 50.

4.3 Результаты работы алгоритмов

На рисунке 4.5 представлен результат работы алгоритма Label Propogation Communities. Всего было выделено 73 сообщества. В текущей визуализации сложно сказать, что алгоритм однозначно хорошо справился со своей задачей, однако он обнаружил малые сообщества, которые мы уже не увидим в следующих алгоритмах.

Рис. 4.5: Визуализация результата работы алгоритма Label Propogation Communities.

На рисунке 4.6 представлен результат работы алгоритма Лювина. Всего было выявлено 16 сообществ. Алгоритм неплохо справился с поставленной задачей, явно можно увидеть 3 выделенных сообщества, а также 4 области с пересекающимися интересами.

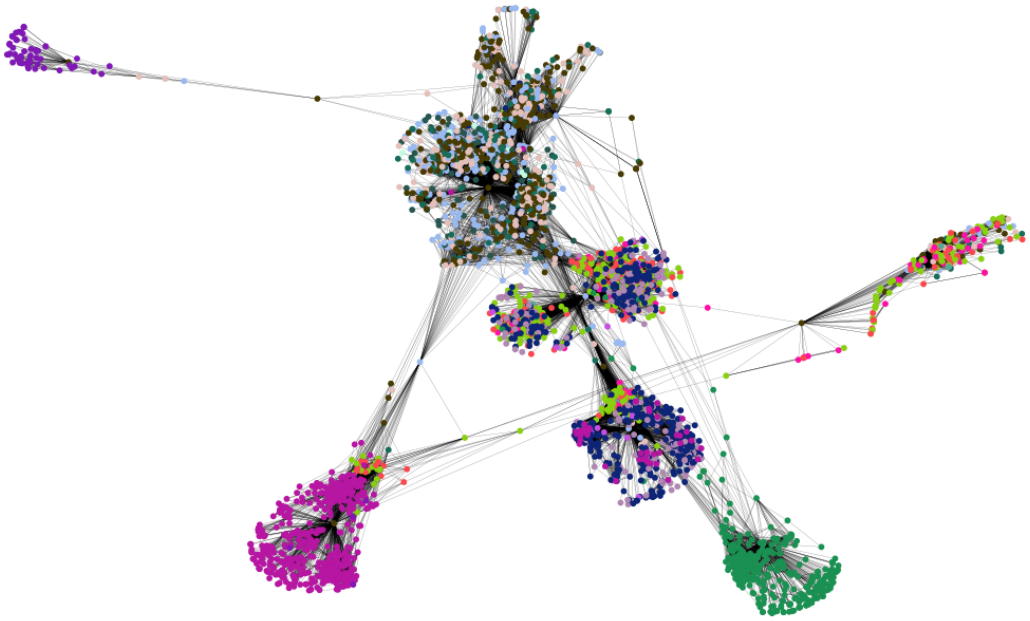

Рис. 4.6: Визуализация результата работы алгоритма Лювина.

На рисунке 4.7 представлен результат работы алгоритма Fluid Communities. Заданное количество сообществ 16.

Рис. 4.7: Визуализация результата работы алгоритма Fluid Communities.

4.4 Зависимость времени исполнения алгоритмов от количества связей в графе

На рисунке 4.8 представлена зависимость времени исполнения алгоритмов LPC, Лювина и Fluid Communities от количества связей в графе.

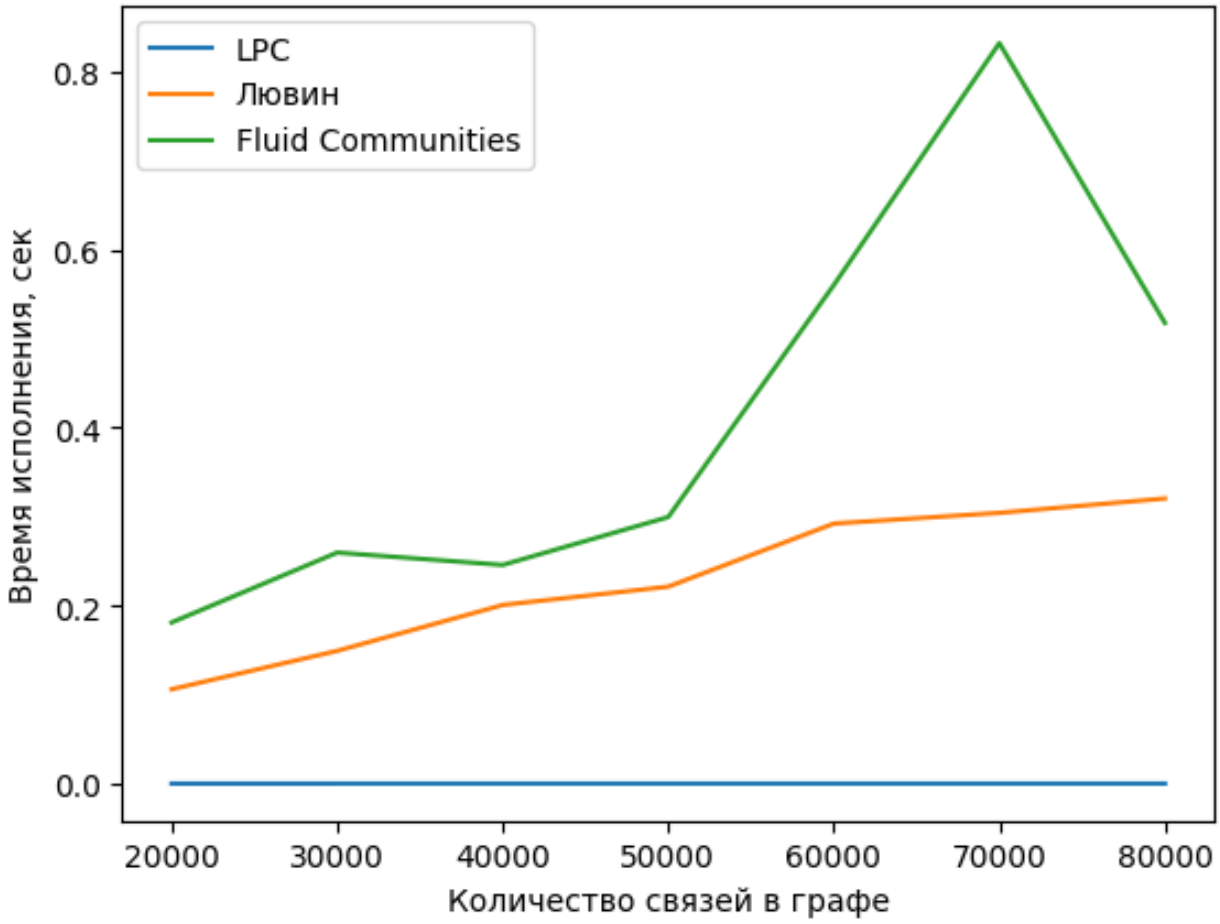

Рис. 4.8: Зависимость времени исполнения алгоритмов от количества связей в графе.

Заключение

В результате проведенных исследований, было получено представление о методах визуализации графов.

Приведенные зависимости времени исполнения алгоритмов от количества связей в графе показало, что даже на достаточно небольшом графе в 80000 связей алгоритмы Лювина и Fluid Communities уже могут исполняться более 0.2 секунд. Кроме того, легко увидеть, что алгоритм Fluid Communities, несмотря на собственное асинхронное исполнение, однозначно проигрывает по времени исполнения и алгоритму Лювина и алгоритму LPC. Кроме того, из графика видно, что алгоритм LPC выполняется на несколько порядков

быстрее двух остальных алгоритмов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была реализована гибридная рекомендательная система.

Были решены следующие задачи:

- приведено описание алгоритма LPC;
- приведено описание алгоритма Лювина;
- приведено описание алгоритма Fluid Communities;
- приведено описание используемых в исследовании данных;
- приведена визуализация результатов;
- проведен анализ скорости работы алгоритмов.

В результате проведенного исследования зависимости времени исполнения алгоритмов от количества связей в графе стало понятно, что даже на достаточно небольшом графе в 80000 связей алгоритмы Лювина и Fluid Communities уже могут исполняться более 0.2 секунд. Кроме того, легко увидеть, что алгоритм Fluid Communities, несмотря на собственное асинхронное исполнение, однозначно проигрывает по времени исполнения и алгоритму Лювина и алгоритму LPC. Кроме того, из графика видно, что алгоритм LPC выполняется на несколько порядков быстрее двух остальных алгоритмов.

Список литературы

1. Snap Stanford [Электронный ресурс]. Режим доступа: http://snap.stanford.edu/data/ego-Facebook.html (дата обращения 26.11.2023).

2. Python official page [Электронный ресурс]. Режим доступа: https://www.python.org/ (дата обращения 10.05.2023).

3. Networkx official documentation [Электронный ресурс]. Режим доступа: https://networkx.org/ (дата обращения 27.11.2023).