

Министерство образования Республики Беларусь  
Учреждение образования  
Белорусский государственный университет  
информатики и радиоэлектроники

УДК 004.415.2

Белов  
Александр Владимирович

Информационная система анализа социальных сетей

### **ДИССЕРТАЦИЯ**

на соискание степени магистра информатики и вычислительной техники  
по специальности 1 - 40 81 02 Технологии виртуализации и облачных  
вычислений

---

Научный руководитель  
Лукашевич Марина Михайловна  
кандидат технических наук, доцент

---

Минск 2018

Нормоконтроль  
Сидорович А.С.

---

# ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....	4
ГЛАВА 1 Обзор литературы .....	7
1.1 Обзор актуальных социальных сетей .....	7
1.2 Обзор аналогов .....	13
1.3 Выбор стека используемых технологий .....	15
1.4 Выводы .....	16
ГЛАВА 2 Теоретические основы рекомендательных систем .....	17
2.1 Коллаборативная фильтрация.....	19
2.2 Метод чередующихся квадратов .....	20
ГЛАВА 3 Проектирование и разработка информационной системы.....	22
3.1 Проектирование модулей .....	22
3.2 Алгоритм получения данных Github.....	25
3.3 Алгоритм получения данных LinkedIn .....	28
3.4 Алгоритм получения данных Facebook .....	30
3.5 Подготовка данных для коллаборативной фильтрации.....	33
3.6 Реализация программной части рекомендательной системы.....	35
3.7 Визуализация данных .....	37
3.8 Описание модели данных.....	42
3.9 Описание структуры и взаимодействия между классами .....	50
ГЛАВА 4 Тестирование информационной системы .....	55
4.1 Автоматизированное тестирование.....	55
4.2 Ручное тестирование.....	56
ЗАКЛЮЧЕНИЕ .....	60
СПИСОК ЛИТЕРАТУРЫ.....	61
ПРИЛОЖЕНИЕ А .....	63
ПРИЛОЖЕНИЕ Б.....	68

## ВВЕДЕНИЕ

Рост популярности социальных сетей, пришедшийся на середину первого десятилетия двадцать первого века, неразрывно связан с развитием веб-сервисов в целом. Сложно перечислить все изменения в социальных отношениях, которые они принесли в нашу жизнь. Социальными сетями различным видов пользуется, по последним данным 2017 года, около трех миллиардов человек на планете Земля. Что составляет значительную часть от четырех миллиардов человек, имеющих постоянный либо периодический доступ к сети Интернет. Несложно сделать вывод, что, в данный момент истории человечества, социальные сети являются наиболее мощной объединяющей силой из всех существующих. Люди из абсолютно разных уголков планеты, имеющие несхожий во многих отношениях менталитет, привычки и потребности, образуют социальные группы и отношения по общим интересам, увлечениям и жизненному опыту.

В процессе постоянного развития, незаметного вначале и все более значительного при приближении к настоящему моменту, социальные сети не оставили неизменной практически ни одну область деятельности человека, имеющую отношение к всеобщей информатизации общества. Профессиональные отношения, сфера образования, электронная коммерция и искусство – области человеческой деятельности, ранее остававшиеся относительно неизменными на протяжении десятилетий, претерпели в кратчайший срок значительные изменения.

Известная в широких кругах профессиональная социальная сеть LinkedIn значительно упростила и расширила область работы с кадрами, внося в отрасль новые стандарты и процессы. Около четырехсот миллионов пользователей имеют активный профиль в сети LinkedIn с актуальной информацией о месте работы, стеке используемых и вызывающих интерес технологий, наградах, сертификатах об окончании курсов и образовании. Для представителей отделов кадров организаций любого размера это означает возможность подбора кандидатов для закрытия конкретной позиции в режиме реального времени, в соответствии с максимально свежими данными о соискателям, потенциальное сокращение бумажного документооборота, связанное с утратой необходимости в ведении бумажных картотек и уменьшение необходимости в личном контакте между сотрудниками организации и соискателями в процессе первоначального исследования рынка вакансий. Информации, доступной для просмотра, обычно бывает достаточно для создания первичного представления о человеке и его степени пригодности для определенной должности. Стоит также отметить отсутствие необходимости сосредотачивать усилия по поиску подходящих кандидатур исключительно на местных специалистах – LinkedIn позволяет

произвести поиск вакансий и кандидатов в международном масштабе. Что, в свою очередь, приводит к повышению интернациональной мобильности населения и увеличению доходов всех заинтересованных сторон.

Несложно догадаться, что с увеличением роли социальных сетей в жизни человека, увеличивается также роль анализа данных, потребность в квалифицированных специалистах в данной области и продвинутых методах анализа, позволяющих обнаружить ранее неизвестные связи между, на первый взгляд, не связанными понятиями и явлениями. И все же, несмотря на постоянно увеличивающиеся объемы данных, доступных для изучения, рост вычислительных мощностей, доступных для широкого круга заинтересованных лиц и совершенствование методов и алгоритмов работы со структурированными и неструктурированными данными, анализ информации и обнаружение знаний остаются нетривиальными задачами, требующими привлечения квалифицированных специалистов и использования алгоритмов, находящихся на переднем крае развития информатики для своего своевременного и корректного решения.

Следует заметить, что предварительным этапом для абсолютно любой задачи, связанной с анализом данных, является сбор данных из релевантных источников. Существуют различные методы, позволяющие получить полезную информацию, пригодную для дальнейшего анализа, из существующих на данный момент социальных сетях. Наиболее удобным с точки зрения исследователя вариантом является анализ открытых наборов данных, содержащих предварительно отобранную и структурированную информацию. Наиболее сознательные игроки рынка, действительно, предоставляют такую возможность. К примеру, для сервиса совместной работы Github в проекте Google BigQuery доступны базы системных событий за достаточно длительные интервалы времени. Что позволяет, в свою очередь, с помощью SQL запроса определенного вида получить интересующую совокупность информационных признаков для любых из имеющихся сущностей системы. Но социальные сети, ориентированные на получение прибыли с помощью рекламы, крайне неохотно предоставляют свои данные в открытом виде, свободном от лицензионных ограничений и судебных претензий. В некоторых случаях, работа с данными сервисами становится возможной благодаря использованию API – системы HTTP запросов определенного вида, служащих для получения и изменения информации сторонними лицами и приложениями. Стоит заметить, что и эта возможность не решает имеющуюся проблему целиком. Практика показывает, что доступная с помощью инструментов разработчика часть данных оказывается малоинтересной для анализа, а наиболее значимая в контексте проводимых исследований часть остается закрытой от доступа общедоступными средствами. Наиболее интересные для анализа данные, такие

как местоположение пользователя, его место работы и обучения, обычно оказывается скрытой от автоматизированных средств сбора данных.

Безусловно, несмотря на все перечисленные сложности, социальные сети являются прекрасным источником данных для научных исследований учреждений образования и коммерческих организаций. Особую значимость в настоящее время приобретает мониторинг сообщений и действий в социальных сетях с целью оценки удовлетворенности пользователя продуктами и услугами, реализации рекомендательных систем и управления процессом выстраивания взаимоотношений между ведущими мировыми компаниями и конечными потребителями.

Несмотря на принципиальную возможность осуществления анализа данных о студентах в международном масштабе, для получения значимых результатов и фокусировки на алгоритмах получения и анализа данных полезным стоит признать ограничение области исследования до более локальных масштабов и, соответственно, построение более специализированного решения.

Задачей разрабатываемой системы анализа социальных сетей является облегчение и автоматизация сбора данных о студентах и выпускниках специальности ВМСиС и дальнейший их анализ с визуализацией результатов на основании публично доступной информации. Следует отметить также важность построения рекомендательной системы, позволяющей анализировать тенденции в действиях пользователей и осуществлять прогнозирование направлений их дальнейших интересов и развития. Информацией, представляющей в отношении пользователей наибольший интерес, является их текущее местоположение, стек используемых технологий, текущее место работы и занимаемые в прошлом должности, вклад в развитие свободного программного обеспечения как оценка общей профессиональности и активности пользователя. На основании перечисленных данных вполне возможно построение процесса пошаговых изменений учебных планов с целью их актуализации и образовательного процесса с целью повышения его качества, доступности и эффективности. Для обеспечения возможности работы с системой не только исследователями и разработчиками, но и сторонними лицами, имеющими отношение к образовательному процессу, является целесообразным проектирование и разработка пользовательского интерфейса на базе современных веб-технологий.

# ГЛАВА 1

## ОБЗОР ЛИТЕРАТУРЫ

В целях определения специфических особенностей строения доступных для анализа социальных сетей, влияющих на доступные методы и протоколы взаимодействия с ними, выяснения положительных и отрицательных сторон каждой, определяющих совокупность информации, в перспективе доступной для анализа, следует рассмотреть общее состояние всей сферы социальных сетей и подобных им сервисов. Также имеет смысл рассмотреть теоретические основы построения рекомендательных систем с целью их интеграции в планируемый программный комплекс и наиболее релевантные на данный момент методы визуализации анализируемых данных.

### 1.1 Обзор актуальных социальных сетей

На основании таблицы 1.1 можно сделать вывод, что наиболее интересными с точки зрения поставленной задачи социальными сетями и сервисами являются LinkedIn и Github. Рассмотрим их подробнее.

Кратко охарактеризовать профессиональную социальную сеть LinkedIn можно как сервис, предназначенный для поиска и установления рабочих контактов в условиях международного рынка, в основном, относящегося к сфере информационных технологий. Следует, однако, заметить, что LinkedIn используется и в сферах, не имеющих прямой связи с информационными технологиями, что является лучшим доказательством неостанавливающегося развития сервиса и увеличения его пользовательской базы. В отличие от стандартных социальных сетей, как всем известные Facebook и Вконтакте, актуальный на пространстве СНГ, для LinkedIn характерен несколько иной подход к идее расширения социального окружения пользователя. Наиболее одобряемым системой способом установления контакта между пользователями является указание точки их профессионального соприкосновения в совокупности с особым сообщением, информирующим адресата о намерениях другой стороны. Таким образом, пользователи должны быть знакомы предварительно, в результате работы в одной и той же организации либо иного способа знакомства на профессиональной почве. Такой подход позволяет достичь точности отображения профессионального круга контактов пользователя, недостижимой для любых других сервисов. Кроме того, в результате стимуляции пользователей к заполнению данных о месте работы, образовании с указанием ученых степеней и применяемых в профессиональной деятельности технологиях, становится доступным анализ данных, которые не представляется возможным получить каким-либо другим способом.

Таблица 1.1 – Сравнительная характеристика социальных сетей

	Facebook	Twitter	Google+	Myspace
	facebook.com	twitter.com	plus.google.com	myspace.com
1	2	3	4	5
Приблизительная месячная аудитория, пользователей	1.59 миллиарда	332 миллиона	418 миллионов	75.9 миллионов
Наличие мобильного приложения	Android, iOS, Nokia, Palm, Blackberry, Windows, Sidekick, Sony	Android, iOS, Blackberry, Windows	Android, iOS	iOS
Интеграция со сторонними сайтами	Есть	Есть	Есть	Есть
Обмен фотографиями	Есть	Есть	Есть	Есть
Обмен видео	Есть	Нет	Есть	Нет
Обмен сообщениями	Есть	Нет	Есть	Есть, через стороннее приложение
Ориентированность на профессиональную аудиторию	Нет	Нет	Нет	Нет
Выручка компании за долларов США	17.928 миллиардов	2.21 миллиарда	74.5 миллиарда	109 миллионов
Число работников	12691	3638	61814	200
Год запуска	2004	2006	2011	2003



Продолжение таблицы 1.1

	Linkedin	Github	Вконтакте	Одноклассники
	linkedin.com	github.com	vk.com	ok.ru
1	6	7	8	9
Приблизительная месячная аудитория, пользователей	400 миллионов	10 миллионов	314 миллионов	200 миллионов
Наличие мобильного приложения	Android, iOS	Android, iOS	Android, iOS	Android, iOS
Интеграция со сторонними сайтами	Есть	Нет	Есть	Есть
Обмен фотографиями	Нет	Нет	Есть	Есть
Обмен видео	Нет	Нет	Есть	Есть
Обмен сообщениями	Есть	Нет	Есть	Есть
Ориентированность на профессиональную аудиторию	Да	Да	Нет	Нет
Выручка компании за 2015 год, долларов США	2.99 миллиарда	Нет данных	90 миллионов	Нет данных
Число работников	8735	467	500	100
Год запуска	2003	2008	2006	2006

Иным доступным для пользователя социальной сети LinkedIn способом расширить свою сеть профессиональных контактов становится взаимодействие со специалистами по подбору персонала, находящихся в процессе поиска специалистов, подходящих для закрытия определенной вакансии. Специалисты по кадрам, так называемые рекрутеры, являются второй из наиболее важных частей аудитории для социальной сети. Специально под них разрабатываются новые инструменты для работы со списками рабочих контактов, требующими, однако, для своей разблокировки, платной подписки и недоступных пользователям социальной сети и ином случае.

Таким образом, становится понятной общая схема монетизации сервиса. Для специалистов, находящихся в поиске рабочего места, существует тарифный план с месячной стоимостью 30 долларов США, который позволяет закрепить свое резюме в верхних строчках результатов поисковой выдачи при фильтрации по определенным технологиям и опыту работы. Также существуют специализированные тарифные планы для руководителей организаций, стоимостью 60 долларов США, с расширенными возможностями по управлению аккаунтами организаций в системе, планы за 80 долларов для специалистов отдела продаж, использующим возможности системы для поиска новых клиентов, а также планы максимальной стоимости в 120 долларов США для рекрутеров, открывающие ранее недоступные возможности по фильтрации резюме кандидатов и позволяющие устанавливать рабочие отношения с соискателями в обход общепринятых механизмов.

Несмотря на то, что для разработчиков доступен API (программный интерфейс), его возможности относительно получения данных выглядят сознательно ограниченными со стороны сервиса. Так, происходит планомерное сокращение доступных методов для манипуляции с данными, доступных для обычных разработчиков, не состоящих в какой-либо крупной организации [1]. Для остающихся независимыми разработчиков остаются доступными методы, относящиеся, скорее, к сфере автоматизации взаимодействия пользователя с ресурсом, как, например, создание нового публично доступного сообщения и подтверждение добавление в контакты. Даже получение информации о профиле пользователя, содержащем данные, относящиеся к профессиональной сфере, ныне не является доступным через API LinkedIn. Несомненно, крупным организациям, как то Samsung, Evernote и китайский WeChat, предоставляется более полный доступ к информации, представляющей интерес.

В результате, при проектировании подсистемы сбора данных, следует учитывать необходимость получения данных, содержащихся в профилях пользователей профессиональной социальной сети LinkedIn и их недоступность через стандартизированные средства доступа. Так как доступ к API ограничен операциями, не относящимися никоим образом к анализу данных,

необходимым этапом является задействование средств анализа веб-страниц, позволяющих автоматизировать действия обычного пользователя социальной сети и получить информацию, содержащуюся в отображенных в результате веб-страницах. Это, безусловно, увеличивает объем необходимой работы, но является единственным приемлемым способом получить подмножество данных, касающихся профессиональной сферы.

Вторым из наиболее интересных для анализа сервисов является система совместной работы Github, использующаяся также как платформа для работы над проектами с открытым программным кодом. Стоит заметить, что среди возможностей, предоставляемых системой Github, также присутствует организация командной работы над проектами и управление процессами и жизненным циклом разработки. Github, безусловно, не является социальной сетью в классическом понимании этого термина, однако может быть с известной долей приближения отнесен к таковым [2]. Присутствует необходимая для классических социальных сетей функциональность обмена видеофайлами, фотографиями, сообщениями, как видно из таблицы 1.1. Действия других пользователей могут быть оценены и прокомментированы. Возможным также является установление рабочих контактов с другими пользователями системы. Таким образом, несмотря на то, что основной задачей системы Github является организация совместной работы над проектами, все остальные возможности социальных сетей в нем также присутствуют, что позволяет использовать данную систему как источник наиболее интересных данных о профессиональной деятельности пользователя.

В отличие от LinkedIn, Github поощряет деятельность независимых разработчиков, направленную на обнаружение новых знаний в генерируемых данных. Сервис имеет наиболее дружелюбное API из всех, доступных для сравнения, а также открытые наборы данных на специализированных платформах. Для незарегистрированного разработчика доступно выполнение 60 запросов стандартного вида с периодом обновления лимита в один час. С момента регистрации в системе и предоставления соответствующих данных в теле запроса, лимиты возрастают до 5000 запросов в час, что уже является достаточным для безостановочной работы цикла по получению данных в одном потоке. При наличии возможности составить расписание запуска задач по импорту данных, ограничение количества запросов и вовсе перестает играть какую-либо заметную роль. Особо стоит отметить, что доступ ко всем возможностям, предоставляемым системой, является бесплатным для разработчиков. Особенностью планов для крупных организаций являются, предположительно, значительно увеличенные лимиты по работе с API, позволяющие производить более частую и полную синхронизацию всех необходимых данных.

Для учащихся и студентов, осуществляющих деятельность в сфере информационных технологий, необходимость в регистрации в системе совместной работы Github возникает либо в процессе обучения в высшем учебном заведении, либо в самом начале трудовой деятельности по специальности. Соответственно, достаточно большое число студентов имеют аккаунты в онлайн системах контроля версий и совместной работы. Наличие актуальных и постоянно обновляемых данных о рабочих проектах пользователя с открытым исходным кодом, используемом стеке технологий и уровню активности в разработке позволяют утверждать, что Github является источником наиболее значительной части информации, связанной непосредственно с рабочим процессом. Также в системе присутствуют бинарные оценки заинтересованности пользователя в проектах иных пользователей – stars. Для подобной информации представляется возможным и важным применение методик предсказания дальнейшего поведения пользователя, что позволит прогнозировать изменение интересов пользователя как разработчика и направление его дальнейшей работы с определенной точностью.

Помимо социальных сетей Github и LinkedIn, которые можно назвать, скорее, профессионально-ориентированными, не следует исключать из рассмотрения социальную сеть Facebook, которая чаще используется в прикладных задачах. Facebook является признанным лидером сферы социальных сетей в том, что касается степени охвата аудитории и точности отображения изменений в социальной жизни пользователей. Таким образом, Facebook не может рассматриваться в качестве достойного доверия источника информации о рабочей информации. Однако, лучшим источником данных о местоположении пользователей не смогла бы стать ни одна другая социальная сеть из рассмотренных. Кроме местоположения, Facebook также имеет смысл использовать для получения списков определенных социальных групп, и для настоящей задачи является наиболее привлекательной возможностью получения списков студентов определенной специальности и учебной группы. Следует также заметить, что полученные данные могут не являться в достаточной степени точными и их следует сверять со сведениями из разного рода официальных источников с целью повышения точности и достоверности результатов.

Что же касается способов получения информации о пользователях социальной сети, Facebook находится по удобству для разработчика в середине условной шкалы между LinkedIn и Github. С помощью API становится возможным получение списка внутренних идентификаторов пользователей, вступивших в определенное сообщество. Однако для самих пользователей доступна лишь самая общая информация, вроде имени и фамилии. Решением этой проблемы является применение методов, схожих с методами получения

данных из социальной сети LinkedIn через автоматизацию работы с веб-страницей. Так становится возможным получение наиболее интересной части информации, которую способна предоставить социальная сеть Facebook – текущее местоположение пользователей с точностью до города.

Для не рассмотренных социальных сетей из таблицы 1.1 следует заметить, что они имеют несколько меньшую полезность для конкретной задачи. Данные, содержащиеся в них, зачастую дублируются в иных сервисах, уже рассмотренных, и получение их из них полезной информации может представлять заметную трудность. Помимо всего прочего, использование социальных сетей регионального уровня, подобных Вконтакте, не является необходимым, поскольку основная масса активных пользователей имеет аккаунты в нескольких социальных сервисах и данные о большей их части можно получить из глобальных социальных сетей. Что делает именно глобальные социальные сети идеальным источником данных для информационной системы.

## **1.2 Обзор аналогов**

Анализ данных социальных сетей, в большинстве случаев, является задачей, решаемой для каждого конкретного проекта индивидуально. Тем не менее, существует ряд сервисов, решающих задачу в общем виде и обладающих возможностями, имеющими достаточно общего с возможностями разрабатываемой информационной системы. Следует перечислить основные из них.

Brandwatch – сервис социальной аналитики, позиционируется как решение для международных компаний благодаря поддержке при анализе 25 языков, в том числе и русского. Применяется для анализа в сферах здравоохранения, спорта, электронной коммерции и демографии. Некоторые клиенты используют Brandwatch для работы с рынком вакансий, что отчасти схоже с задачами LinkedIn. Плюсом сервиса является доступность для анализа данных, накопленных с 2010 года, что зачастую бывает крайне необходимо для понимания тенденций. Основная часть кода, относящегося непосредственно к аналитике, написана на языке Java с использованием NoSQL баз данных. Визуализация данных реализована с использованием веб-технологий на базе Javascript библиотек визуализации данных [3].

33Across – позиционирует себя как платформа для анализа социального трафика и монетизации контента. Стек технологий включает в себя Java, Ruby, Ruby on Rails, Hadoop. Возможным минусом следует считать узкую направленность сервиса, однако, фокусирование на одном направлении позволяет достичь в нем лучших результатов [4].

Hootsuite – аналитика социальной активности. По сути, позволяет оценивать тональность реакции аудитории на публикации представителей организаций-заказчиков. В разработке используется Python, нереляционные базы данных и хранилища ключ-значения, к примеру, Redis и Memcached. Аналитическая часть использует Scala и систему передачи сообщений Akka [5].

Moz – оценка влияния поведения компании в социальных сетях на результаты поискового продвижения (SEO). Использует Node.js, NoSQL базы и кластерную обработку данных. Имеет отличный модуль визуализации, однако, анализ данных весьма узконаправленный [6].

Так как в разрабатываемой информационной системе планируется использование рекомендательных систем, следует также перечислить ряд сервисов данного направления.

Прежде всего, рекомендательная система реализована самим сервисом Github, что позволяет, учитывая различные метрики и поведение пользователя, рекомендовать ему проекты для дальнейшего изучения. Несмотря на отсутствие достоверной информации об используемом технологическом стеке, скорее всего, используется язык Ruby и интерфейсы для низкоуровневых библиотек, написанных на C, с целью достижения наилучшего быстродействия. Используемые алгоритмы сходны для всех рекомендательных сервисов.

Многие разработчики, заинтересованные в изучении рекомендательных систем, выбирают предметной областью именно репозитории системы Github, благодаря доступности данных в открытом виде и общей пригодности данной области для построения рекомендательных систем по методу коллаборативной фильтрации. Среди множества особо стоит отметить два.

Gazer – написанная более пяти лет назад рекомендация система, не использующая коллаборативную фильтрацию [7]. Вместо нее была задействована подобранная экспериментально методика расчета схожести двух репозиторий на основании процента пересекающихся оценок пользователей. Весь основной функционал реализован на языке Javascript и имеет веб-версию, все еще функционирующую. Следует заметить, что результаты не отличаются особой точностью и могут быть устаревшими.

GHRecommender – более новая система, реализованная в сентябре 2017 года двумя российскими программистами [8]. В отличие от Gazer, использует классическую коллаборативную фильтрацию и более того, основным используемым методом является метод чередующихся наименьших квадратов. Веб-часть написана на Javascript с использованием React и Semantic UI. Логика на стороне сервера реализована на языке Python с использованием фреймворка Django. Данные для анализа получаются с помощью сервиса GHTorrent, что является неплохой альтернативой использованию с той же целью API и открытых наборов данных.

### 1.3 Выбор стека используемых технологий

Из рассмотренных аналогов следует сделать вывод о типичном для отрасли технологическом стеке.

Для хранения данных практически в каждом из сервисов используются нереляционные базы данных. Выбор в пользу именно этого типа хранилища сделан из-за невозможности полноценно отобразить собранные данные на реляционную модель без времязатратной предварительной обработки. Однако в паре с NoSQL базами используются также и обычные реляционные базы данных, которые значительно лучше подходят для хранения результата аналитики.

Тем не менее, использование нереляционных баз данных не планируется. Сфера применения NoSQL решений и стандартных реляционных баз данных в значительной степени совпадают. А значит, имеет смысл выбрать в качестве предпочтительной базы данных одно из проверенных временем универсальных решений, например, PostgreSQL, который также имеет некоторые возможности, свойственные нереляционным базам данных.

Визуализация данных в основном реализуется с помощью библиотек визуализации данных на основе JavaScript, например, d3js. Впрочем, это характерно для динамической инфографики в целом, JavaScript является гораздо более предпочтительным решением относительно системных библиотек визуализации данных.

Для разрабатываемой информационной системы не ставятся жестких ограничений по времени сбора и обработки информации. С учетом указанных факторов наиболее подходящим является язык высокого уровня Ruby. Ruby – интерпретируемый язык программирования высокого уровня. Обладает независимой от операционной системы реализацией многопоточности, строгой динамической типизацией, «сборщиком мусора» и многими другими возможностями, поддерживающими много разных парадигм программирования. Ruby был задуман в 1993 японцем Юкиhiro Мацумото, стремившимся создать язык, совмещающий все качества других языков, способствующие облегчению труда программиста [9].

К основным достоинствам языка Ruby можно отнести открытый исходный код [10]. Преимущество такого рода программ является возможность просмотра, изучения и изменения, что позволяет любому желающему принять участие в доработке самой открытой программы, использовать код для создания новых программ и исправления в них ошибок.

Важной чертой Ruby является гибкая возможность метапрограммирования. Тот факт, что Ruby – интерпретируемый язык, позволяет создавать выполняемый код программным образом, что значительно

сокращает время реализации сложного функционала и благоприятно влияет на стоимость разработки проекта в целом.

Также следует отметить использование в языке динамической типизации [11]. Этот приём широко используется в языках программирования и позволяет связывать переменную с типом в момент присваивания значения, а не в момент объявления переменной. Таким образом, в различных участках программы одна и та же переменная может принимать значения разных типов.

Сторонние библиотеки, которые носят название *gem*, делают Ruby уникальным инструментом с огромными возможностями. На данный момент доступно более 2.5 миллионов различных библиотек [12]. Таким образом, среди всего разнообразия бесплатных и открытых библиотек с большой долей вероятности можно найти решение вашей текущей проблемы, что значительно ускоряет скорость разработки.

Однако для построения рекомендательных систем обычно используется язык Python, предоставляющий значительно лучшие возможности по работе с данными. Наличие библиотек *scikit-learn*, *pandas*, *scipy* делает его практически незаменимым для данной задачи. Так как основная часть приложения реализована на языке Ruby, задачи, требующие использования языка Python, вынесены в отдельные скрипты, запускаемые встроенными средствами языка Ruby.

Таким образом, выбранные технологии покрывают все основные запросы проекта и позволяют завершить его разработку с привлечением минимально возможного количества ресурсов и в минимальный срок.

## **1.4 Выводы**

На основании анализа доступных аналогов можно сделать вывод, что большая часть доступных систем значительно усложнена с целью обеспечить покрытие более широкого спектра задач. Было решено создать информационную систему, которая позволит решить проблему анализа социальных сетей без необходимости использования кластерных систем за адекватное время. В качестве исследуемой социальной группы выбраны студенты специальности ВМСиС. Основной функционал системы включает в себя:

- сбор данных из социальных сетей, признанных наиболее подходящих для решения конкретной задачи;
- анализ собранных данных с помощью построения доступных для восприятия графиков и рекомендательных систем;
- формирование отчетов и визуализация результатов.



## ГЛАВА 2

# ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМ

Особый интерес для анализа представляют данные, полученные с помощью API сервиса совместной работы Github. Помимо очевидных данных, доступных для непосредственного анализа, таких как используемые в проектах программиста языки, популярность программных проектов и число подписчиков, интерес так же представляют проекты, в отслеживании прогресса которых пользователь был заинтересован. В терминах Github это starred репозитории, на обновления которых можно подписаться.

Прежде всего, данные репозитории отражают область профессиональных интересов разработчика. Соответственно, приоритетной является задача анализа существующих интересов пользователя и подбор рекомендаций для дальнейшего изучения на их основе. Задачи подобного рода решаются рекомендательными системами четырех типов: коллаборативная фильтрация, фильтрация на основе содержания, фильтрация, основанная на знаниях и гибридные системы. Рассмотрим данные четыре типа рекомендательных систем с целью определить наиболее подходящий для задачи анализа пользовательских репозиторий Github.

Коллаборативная фильтрация основывает свои рекомендации на поведении пользователя в прошлом. Следует заметить, что обучение системы возможно на данных о поведении одного пользователя, но для наиболее эффективной работы модели требуется учитывать также поведение других пользователей со сходными характеристиками. К плюсам данной модели можно отнести ее относительную простоту и, при этом, высокую точность. Минус так или иначе проявляется во всех типах рекомендательных систем – это проблема холодного старта. Для эффективного анализа и подбора рекомендаций мы должны обладать достаточно полными данными о поведении пользователя в течении продолжительного времени. Для новых пользователей с неизвестными предпочтениями подбор рекомендаций невозможен либо сильно затруднен.

Фильтрация, основанная на контенте (content-based), подбирает рекомендации исходя из предыдущих оценок пользователя для сходных групп товаров. Анализ в данном случае является более формальным, однако необходимым свойством объекта исследований является наличие большого количества информационных полей, обеспечивающих возможность классификации по широкому набору признаков. Плюсом данной системы является менее заметная проблема холодного старта, что упрощает составление рекомендаций для пользователей с недостаточной информацией об их прошлом

поведении. Также возможным становится рекомендация объектов, не оцененных еще ни одним пользователем, на основании их набора признаков. Минусом является меньшая, по сравнению с коллаборативной фильтрацией, точность и существенно более сложная реализация, за счет необходимости дополнительного анализа наиболее эффективных признаков и, в некоторых случаях, уменьшения размерности данных. Данный тип рекомендательных систем часто используется в сервисах рекомендаций книг и фильмов, например, kinopoisk и livelib. Объясняется это, прежде всего, особенностью рекомендуемых объектов. И для книг, и для фильмов существует значительное количество признаков, на основании которых может производиться фильтрация: жанр, год выпуска, объем и тип произведения.

Для рекомендательных систем, основанных на знаниях (knowledge-based) характерна еще большая сложность в разработке и поддержке. Фильтрация, основанная на контенте, является частным случаем систем, построенных на знаниях, но имеет значительную распространенность, что позволяет вынести ее в отдельный тип. Соответственно, для рассматриваемого типа систем характерны сложные условия фильтрации и знания, извлеченные из различных метрик. Плюсом является возможность более интеллектуального составления рекомендаций, с исключением результатов, более не актуальных для конкретного пользователя и результатов, не соответствующих предыдущим предпочтениям, но, с высокой степенью вероятности, могущих заинтересовать пользователя. Минусом является самая высокая сложность разработки, сбора и подготовки данных.

Гибридные системы комбинируют все перечисленные подходы, что позволяет избежать ограничений, свойственной каждой отдельной модели. При этом, все же, минусом является более высокая сложность разработки в сравнении с коллаборативной фильтрацией.

Для того, чтобы сделать выбор типа рекомендательной системы, следует также учитывать особенности исследуемых данных. Для рассматриваемых репозиториях пользователя в системе Github характерным признаком является отсутствие явных признаков, позволяющих использовать фильтрацию, основанную на контенте. Данных о языках программирования, используемых в конкретном репозитории, абсолютно недостаточно для построения эффективных рекомендаций. Подходом, имеющим хорошие перспективы, является семантический анализ исходного кода проектов, позволяющих выделить общие признаки в синтаксисе и разметке, использовании определенных возможностей языка и библиотек. Данный подход можно отнести к системам, основанным на знаниях, и заметно, что он является весьма непростым в реализации и обладает повышенными требованиями к фазе первоначального сбора данных. Подход, основанный на коллаборативной

фильтрации, является оптимальным для составления рекомендаций по репозиториям Github, так как информацию можно представить в виде пар пользователь и заинтересовавший его репозиторий, что в дальнейшем может быть представлено в виде разреженной матрицы. Рассмотрим коллаборативную фильтрацию более подробно.

## **2.1 Коллаборативная фильтрация**

Пользователи, оценившие определенные объекты в прошлом, склонны давать сходные оценки и впоследствии. На данном предположении, неоднократно исследованном и подтвержденном [13], основываются рекомендательные системы по методу коллаборативной фильтрации. Для коллаборативной фильтрации также принято деление на типы методов: методы, основанные на соседстве, на модели и гибридные.

Методы, основанные на соседстве, были разработаны первыми и являются наиболее простыми. Тем не менее, они используются во многих рекомендательных системах. Основной принцип данного метода — рекомендации составляются путем соотнесения пользователя с подгруппами схожих пользователей.

Методы, основанные на модели стали доступны позднее, и используют они многие из доступных средств машинного обучения, методы кластеризации, вероятностный анализ, марковские цепи. В целом, подход данный метод дает более точные прогнозы и помогает обнаружить скрытые факторы, объясняющие наблюдаемые оценки пользователей. Также метод на модели лучше масштабируется для больших наборов данных и лучше оперирует разреженными матрицами. Недостаток очевиден, создание модели не является простым процессом, для этого необходим всесторонний анализ входных данных.

Гибридный подход объединяет два выше перечисленных и является в настоящий момент самым распространенным. Он же является и самым сложным в разработке и дорогим в применении. Однако использование гибридных методов позволяет избавиться от недостатков классических методов и сохранить их преимущества.

Конкретно для задачи создания рекомендаций по репозиториям программных проектов на Github имеет смысл рассматривать, прежде всего, методы, основанные на соседстве, поскольку в данном случае просто получить доступ к большой базе оценок. Простота данного метода также играет роль, для достаточно больших объемов обучающей выборки более простые алгоритмы в данных задачах зачастую показывают лучшее время обучения и достаточно корректные результаты.

Для методов коллаборативной фильтрации, основанных на соседстве, важную роль играет выбор меры схожести, которая представляет собой метод подсчета расстояния между векторами в  $N$ -мерном пространстве. Возможных вариантов достаточно много, среди них косинусная мера, коэффициент Пирсона, Евклидово расстояние, коэффициент Танимото и многие другие. Наиболее часто используемыми среди них являются косинусная мера и коэффициент Танимото. Среди более свежих разработок стоит отметить метод чередующихся наименьших квадратов (ALS). Наличие библиотеки `implicit` для языка Python, позволяющей использовать ускорение на GPU делает его достаточно интересным выбором. Так как целью настоящей работы является анализ данных именно репозитория Github, метод ALS предпочтителен, поскольку работает именно с наборами объектов пользователь и уже опробован в задачах подобного рода [14].

## 2.2 Метод чередующихся квадратов

Метод чередующихся наименьших квадратов (Alternating Least Squares, ALS) является простым, но в то же время весьма мощным способом решения задач, связанных с разработкой коллаборативных рекомендательных систем, основанных на методе соседства. Представленный алгоритм работает с разреженными матрицами, имеющими значительное количество нулевых элементов при достаточно большой размерности. Таким образом, для представленных данных матрица  $A$  состоит из данных о пользователе и репозиториях, отмеченных им. Единичное значение в строке  $i$  и столбце  $j$  матрицы означает, что пользователь  $i$  отметил репозиторий  $j$ , как интересующий его. Алгоритм чередующихся наименьших квадратов факторизует матрицу  $A$  как произведение двух плотных матриц  $X$  и  $Y^T$  меньшей размерности, как показано на рисунке 2.1 [15].

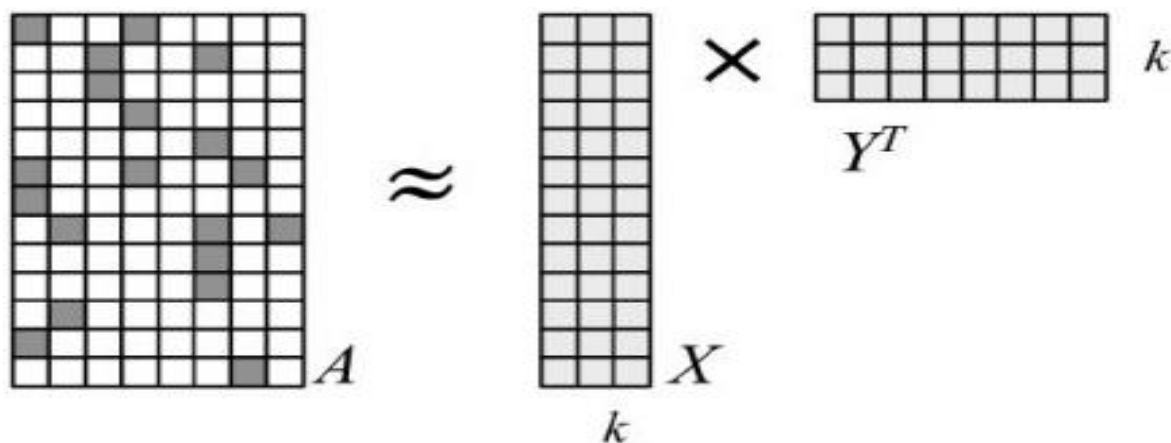


Рисунок 2.1 – Матричное представление данных для метода ALS

Столбцы  $k$  в данных матрицах соответствуют скрытым свойствам, которых позволяют определить скрытые данные о взаимодействиях между пользователями и выбранными репозиториями. Факторизация является лишь приближительной, поскольку произведение матриц  $X$  и  $Y^T$  будет являться значительно более плотным, чем исходная матрица  $A$ . Стоит заметить, что матрицу  $X$  можно определить, как матрицу взаимодействия «пользователь – свойство», а матрицу  $Y$  – «свойство – репозиторий». Алгоритм чередующихся наименьших квадратов вычисляет значения матриц  $X$  и  $Y$ .

$$A_i \cdot Y \cdot (Y^T \cdot Y)^{-1} = X_i \quad (2.1)$$

Задачей алгоритма ALS является уменьшение квадрата разности между строковыми векторам, вычисляемыми по формуле (2.1), в соответствии с формулой (2.2).

$$|A^i Y (Y^T Y)^{-1} - X_i| \quad (2.2)$$

Матрица  $Y$  неизвестна, но может быть инициализирована случайными числами, в данном случае при выполнении алгоритма мы сможем получить первичные значения матрицы  $X$ . Алгоритм, заключенный в формуле (2.1) в достаточной степени подходит для распараллеливания, поскольку каждая строка матрицы  $X$  может быть рассчитана отдельно от остальных на основании матрицы  $Y$  и одной из строк исходной матрицы  $A$ .

Затем такие же вычисления применяются для подсчета строк матрицы  $Y$  исходя из доступной матрицы  $X$ . После чего матрицы снова меняются местами и так до получения приемлемого решения. Таким образом, метод ALS основан на метрике наименьших квадратов и является чередующимся.

## **ГЛАВА 3**

# **ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ**

Для достижения возможности выборочной модернизации отдельных составных частей проекта без необходимости изменения остальных, информационную систему следует разделить на логически взаимосвязанные модули. В первую очередь, для определения перечня необходимых модулей, следует определить основные возможности информационной системы:

- возможность авторизации в социальных сетях, определенных ранее;
- возможность сбора данных из социальных сетей;
- анализ собранных данных различными способами;
- дополнение собранных данных о местоположении пользователя географическими координатами;
- возможность импорта дополнительных данных для облегчения задачи анализа;
- экспорт полученных результатов для дальнейшего использования;
- визуализация данных путем генерации таблиц и графиков;
- формирование отчетов по результатам анализа данных.

### **3.1 Проектирование модулей**

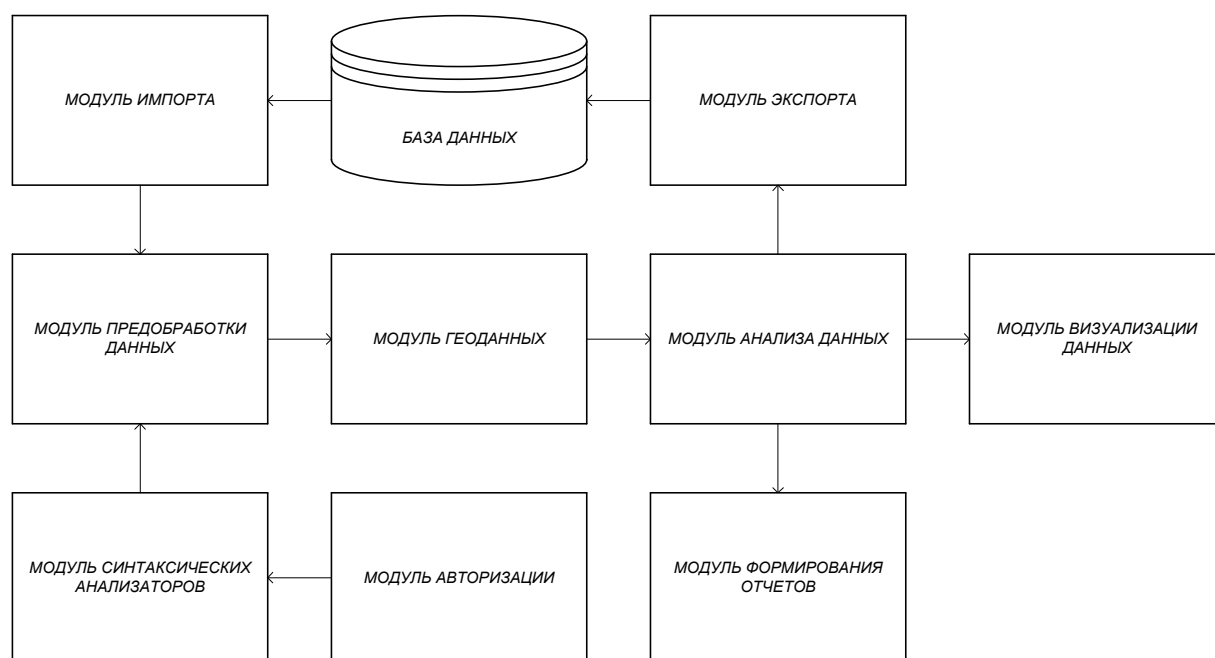
Для решения каждой из вышеперечисленных задач или их комбинации следует выделить один логический модуль. Таким образом, задачи будут транслированы в систему модулей однозначным образом. Связь каждого модуля с одним или несколькими другими модулями позволит обеспечить работоспособность системы как единого целого. Как правило, связь между модулями реализуется через взаимное использование алгоритмов и обмен данными. Обобщенная структурная схема информационной системы изображена на рисунке 3.1.

Из выделенных на представленной структурной схеме десяти модулей, критически важными являются модули синтаксических анализаторов и анализа данных. Оставшиеся модули, решающие задачи предобработки и постобработки данных, импорт и экспорт, не являются критичными для функционирования основных подсистем, однако важны для безошибочной работы системы в целом.

*База данных* является основным местом хранения информации, полученной из рассмотренных социальных сетей, а также для результатов анализа данных и промежуточных результатов. Является внешним модулем по отношению к всей информационной системе. Как было определено в разделе

1.3 диссертации, в качестве базы данных следует использовать PostgreSQL [16]. Основными преимуществами данного решения являются:

- практически неограниченный размер хранилища;
- механизмы транзакций и репликации;
- возможность использования особых форматов для географических координат и наличие специализированных функций для работы с ними;
- широкие возможности масштабирования.



**Рисунок 3.1 – Обобщенная структурная схема**

Основным назначением *Модуля импорта* является обработка загруженной дополнительной информации. Так, при анализе данных студентов, дополнительной информацией может являться список выпускников за различные годы обучения, так как получение этих данных из других источников не может обеспечить приемлемой точности. Форматом предоставления данных для рассматриваемого модуля является формат значений с разделением запятыми (CSV), который является наиболее простым и быстрым в обработке, пригоден для редактирования с помощью широко распространенного офисного пакета Microsoft, а также обладает отлично подходит для представления данных в проектируемой информационной системе.

Как можно понять из названия, *Модуль преобработки данных* реализует методы преобразования в вид, пригодный для дальнейшего анализа, данных, полученных из социальных сетей Facebook и LinkedIn. Формат сырых данных для выше перечисленных социальных сетей представляет из себя теги HTML,

извлеченные из ответа серверов систем и слабо поддается анализу в неподготовленном виде. Извлечение и группировка информация позволяет применить и к этим данным классические способы анализа, что и является основным назначением модуля.

*Модуль геоданных* служит для обработки информации о географическом положении пользователей социальной сети Facebook и системы LinkedIn. Так как информация в своей изначальной форме представляет собой лишь названия городов, это не позволяет доступными средствами оценить распределение пользователей на мировой карте. Однако из названия города становится возможным получить с помощью существующих сервисов геокодинга пары географических координат, широту и долготу, которые, в свою очередь, позволяют точно определить страну и континент, а также использовать специализированные функции PostgreSQL по работе с географическими данными.

*Модуль авторизации* предоставляет возможность авторизации в веб-интерфейсе и API социальных сетей. Авторизация через API является, в большинстве случаев, достаточно простой задачей, однако авторизация через веб-интерфейс требует автоматизации действий пользователя, выполняемых через веб-браузер, что позволяет в дальнейшем использовать веб-страницы, доступные лишь авторизованным пользователям, при дальнейшем анализе в модуле синтаксических анализаторов.

Одной из наиболее сложных и критически важных частей проекта, как уже было сказано, является *Модуль синтаксических анализаторов*. Единственным способом получения данных из социальных сетей, не поддерживающих API для разработчиков, либо не реализующих в API все необходимые методы, является синтаксический анализ веб-страниц. Данная практика осуждается со стороны таких компаний, как LinkedIn, однако является в ряде случаев единственным возможным выходом. С точки зрения социальной сети происходит просмотр пользователем веб-страниц, однако на самом деле выполняет все необходимые действия скрипт и в процессе его работы происходит анализ ответов сервера с целью выделения фрагментов информации. Формат получаемых данных для различных социальных сетей в значительной степени отличается, что приводит к необходимости использовать различные синтаксические анализаторы для каждого нового случая.

Конечной целью получения данных и их предобработки является анализ, выполняемый вторым по важности модулем проекта – *Модулем анализа данных*. С использованием определенных математических методов и алгоритмов проводится извлечение из подготовленных данных знаний. Например, статистика по навыкам, местам работы и образованию может быть получена путем анализа данных профессиональной социальной сети LinkedIn, а



для информации об используемых языках программирования и работе в области проектов с открытым исходным кодом следует использовать Github. Именно совокупный анализ данных, а также использование рекомендательных систем, позволяет выделить знания из полученной информации.

Для экспорта во внешние системы данных в различных форматах предназначен *Модуль экспорта*. Как и в случае с модулем импорта, наиболее удобным форматом для экспорта является CSV, который подходит для более упорядоченных результатов анализа, хорошо описываемых реляционной моделью, куда больше, чем для хаотичных импортируемых данных.

*Модуль оформления отчетов*. Экспорт данных служит для вывода сырых данных, полученных в результате анализа, однако экспортируемые данные больше подходят для использования во внешних программных системах. Для ознакомления же пользователя с результатами анализа служит модуль оформления отчетов, который позволяет оформлять результаты в виде, пригодном для рассмотрения человеком. Предпочтительным форматом для создания отчета является формат PDF. Отчеты содержат результаты анализа, отформатированные и снабженные графическими данными.

*Модуль визуализации данных* служит для генерации графиков и таблиц различных видов на основании результатов анализа. Полученные графики следует использовать для отображения результатов в интерфейсе информационной системы, а также, в некоторых случаях, при составлении отчетов.

### **3.2 Алгоритм получения данных Github**

Перед анализом алгоритмов, направленных на сбор данных более сложными и менее одобряемыми компаниями способами, имеет смысл начать с рассмотрения работы с классическим API сервиса совместной работы Github, для использования которого не придется нарушать условия лицензионного соглашения, как в случае с социальными сетями LinkedIn и Facebook.

Сервис Github размещает абсолютно всю необходимую в настоящем проекте информацию в публично доступном виде. Сервис создан разработчиками для разработчиков и в этом состоит его основное отличие от иных социальных сетей, вроде Facebook и LinkedIn. Программный интерфейс Github является бесплатным лишь отчасти, однако предоставляет достаточно возможностей без внесения какой-либо платы [17]. Ограничивается лишь количество запросов за определенный период времени, что является очень удобным вариантом для исследователей, которым не принципиально немедленное получение большого объема данных и приложения сбора информации которых поддерживают техническую возможность запуска задач

по расписанию. Также среди всех вариантов внешних API, используемых в настоящей диссертации, API Github также является наиболее логично структурированным среди всех API, используемых в настоящей диссертации, и реализует именно то поведение, которое ожидает разработчик.

Доступность библиотеки Octokit для языка Ruby, реализующей программный интерфейс Github и получающей постоянные обновления, также является весьма важным преимуществом [18]. Стандартным видом представления API является описание протокола взаимодействия через HTTP-запросы, что является не слишком удобным способом для применения в объектно-ориентированном программном продукте. Объектно-ориентированная природа программного продукта делает не слишком удобным использование стандартно предоставляемого API, описанного как набор HTTP-запросов для взаимодействия с сервисом. Библиотека Octokit, в то же время, предоставляет возможность работы с программным интерфейсом Github с использованием объектно-ориентированного подхода, в котором сущности предоставляются в виде объектов либо структур с самоочевидными названиями полей и методов.

Для начала работы с библиотекой Octokit следует указать логин и пароль от аккаунта на Github, что позволит поднять лимит на запросы с 60 до 5000 запросов в час. Это довольно значительная величина, такого количества запросов вполне достаточно для решения всех задач, связанных непосредственно с сервисом совместной работы Github. Особо следует отметить, что, для наиболее полного использования выше названного лимита операций, следует производить работу с сервисом Github в одном потоке. Инициализация библиотеки производится следующим образом:

```
Octokit.configure do |c|  
  c.login = Rails.application.secrets.github_login  
  c.password = Rails.application.secrets.github_password  
end
```

Как заметно из данного примера, инициализация производится путем передачи в метод класса блока кода с присвоением переменным значений логина и пароля. Важным моментом, на который следует обратить особое внимание, является то, что всю приватную информацию, такую как логины и пароли от использованных сервисов, следует выносить в отдельный файл, не отслеживаемый в системе контроля версий.

На момент использования Octokit с целью получения данных о пользователях в сервисе совместной работы Github с системе, как правило, уже присутствует информация о пользователях, полученная иными методами.

Таким образом, поиск пользователя Github проводится на основании известного полного имени импортированного пользователя. Метод `search_user`, принимающий на вход полное транслитерированное имя пользователя с запускающий процесс анализа данных Github, демонстрирует данный подход:

```
def search_user(name)
  logins=Octokit.search_users(name).items.map(&:login)
  @users=logins.map{|login| Octokit.user(login)}
  save_user
end
```

Следует заметить, что в приведенном выше программном коде используются методы, характерные для функционального программирования. Следующим этапом является сохранение информации о найденных пользователях, на этапе которого и проявляется вся лаконичность работы с объектным отображением программного интерфейса Github.

```
def save_user
  return if @users.empty?
  puts "Found Github users".blue
  @users.each do |user|
    github_data = GithubData.create(
      login: user.login,
      name: user.name,
      email: user.email,
      num_repos: user.public_repos,
      num_followers: user.followers,
      num_following: user.following,
      blog: user.blog
    )
    if user.avatar_url.present?
      puts "Saving avatar image".yellow
      Image.create(
        imageable: github_data.user,
        link: user.avatar_url
      )
    end
    save_repositories(user.login)
  end
end
```

Таким образом, все действия разработчика при работе с предоставляемым программным интерфейсом сводятся к вызову методов, предоставляемых библиотекой Octokit для объектов данных.

Программный код для сохранения информации о репозиториях и языках программирования также представляет интерес, ознакомиться с ним можно в приложении А к настоящей пояснительной записке. Описанный в приложении код повторяет основные принципы работы с программным интерфейсом Github на примере другого типа данных.

### 3.3 Алгоритм получения данных LinkedIn

Высокая степень открытости данных в социальной сети значительно облегчает работу с ее данными, что было продемонстрировано в подразделе 3.2 на примере алгоритма работы с сервисом Github. Социальные сети LinkedIn и Facebook обладают значительно меньшей открытостью и, для получения наиболее интересных с точки зрения анализа данных, появляется необходимость использовать нестандартные способы, идущие в разрез с официальной политикой пользования системой. При определенном стечении обстоятельств, они могут привести к блокировке аккаунта и даже, как показала практика, к судебным искам. Рассмотрим данную ситуацию на примере профессиональной социальной сети LinkedIn.

```
def get_public_member(name)
  @public_profile = @web_scraper.scrape(name)
  return if @public_profile.nil?
  @profile = begin
    Linkedin::Profile.new(
      "linkedin.com/in/#{@public_profile}",
      {company_details: true}
    )
  rescue
    nil
  end
  save_public_member
end
```

Как несложно заметить из приведенного программного кода, в случае с LinkedIn также используется сторонняя библиотека [19]. Упомянутая библиотека, в данном конкретном случае, служит для получения публично доступного профиля пользователя с целью дальнейшего его анализа. Однако

для получения публичного профиля в сети LinkedIn необходимо знать ссылку на публичный профиль пользователя, найденного по полному имени и для выполнения этой задачи используется библиотека Mechanize, предназначенная для эмуляции взаимодействия пользователя с веб-интерфейсом LinkedIn. Библиотекой реализуется специальный класс `LinkedinWebScraper`, основные методы которого рассмотрены ниже:

```
def scrape(name)
  authorization if !@logged_in
  url = "?type=people&keywords=#{
    name.split(' ').join('+')
  }"
  @results = @agent.
    get(url).
    body.
    scan(/\{"person"\: \{.*?\}\}/)
  return if !@results[0].present?
  @profile_id = begin
    JSON.parse(@results[0]).
      to_s.
      scan(/profile\/view?id=(.*?)&/).
      compact.first.first
  rescue
    nil
  end
  get_public_profile
end

def get_public_profile
  url = "?id=#{@profile_id}"
  profile_raw = @agent.get(url).body
  @public_profile = begin
    profile_raw.scan(/\.com\/in\/(.*?)\/\//).first.first
  rescue
    nil
  end
end
```

Полную версию кода класса, описываемого выше, можно изучить в приложении А к настоящей диссертации. Несложно заметить, что работа с API

является значительно более простой в плане разработки, чем эмуляция взаимодействия пользователя с веб-интерфейсом. Для данного подхода характерно частое использование в коде алгоритма регулярных выражений, которые являются сложным для тестирования и понимания методом. Также следует отметить, что для работы с веб-страницами многие части алгоритма должны быть снабжены блоками обработки ошибок, поскольку работа с сервисами через автоматизацию действий пользователя отличается довольно низкой надежностью и предсказуемостью, а также высоким процентом ошибок, происходящих в ходе задействования в автоматическом режиме веб-браузера.

Задачей, решаемой выше описанным кодом, является поиск пользователя в системе LinkedIn по его полному транслитерированному имени, а также получение ссылки на публично доступный профиль для его дальнейшего анализа. После получения ссылки на профиль становится возможным использование библиотеки LinkedInScraper, служащей для получения данных о местах работы, навыках и иных доступных данных пользователя. Подход, используемый внутри библиотеки, является аналогичным варианту с эмуляцией действий пользователя в браузере, однако библиотекой предоставляется в значительной степени более комфортный интерфейс, делающий работу с профилями проще и понятнее, чем те же самые действия, выполняемые с применением анализатора страниц Mechanize.

Таким образом, следует заметить, что в процессе сбора данных профессиональной социальной сети LinkedIn API, предоставляемое компанией, не используется совершенно. Вся работа с данными выполняется посредством использования парсеров веб-страниц и является, по большому счету, нарушением пользовательского соглашения в отдельных его пунктах. Летом 2017 года компания LinkedIn пыталась запретить разработчикам анализ публично доступных данных, однако, по предписанию суда, вынуждена была разрешить это делать. Тем не менее, подход не является официально одобряемым.

### **3.4 Алгоритм получения данных Facebook**

В определенной степени гибридом между подходами LinkedIn и Github является алгоритм получения данных из социальной сети Facebook. Для Facebook доступна достаточно удобная в использовании библиотека Koala, написанная на языке Ruby [20]. Однако из-за ориентированности библиотеки на данные, представляющие значительную сложность для анализа в результате использования естественного языка и являющиеся незначительной частью общего объема данных, доступной социальной сети Facebook, для исследователя работа с Facebook даже через библиотеку Koala представляет

определенную трудность. Для настоящего проекта было принято решение о необходимости получения данных о местоположении пользователей социальной сети любой ценой, поскольку именно пользователи социальной сети Facebook максимально точно идентифицируются со специальностью ВМСиС и содержат данные о местонахождении в обязательном порядке.

```
def get_group_members
  @graph = Koala::Facebook::API.new(token)
  puts "Getting group #{@gid} members...".blue
  begin
    @members = @graph.get_connections(@gid, "members")
  rescue
    puts "Refresh token!".red
  end
end

def save_group_members
  self.get_group_members if !@members.present?
  return if @members.empty?
  @members.each do |member|
    facebook_data = FacebookData.create(
      original_name: member["name"],
      facebook_id: member["id"]
    )
    puts "Save fb user ".blue
  end
end
```

Список пользователей определенной группы в социальной сети Facebook, как заметно из представленного алгоритма, получается с использованием библиотеки Koala. Но единственной информацией о пользователях, содержащейся в ответе социальной сети, являются полное имя пользователя, представляющее определенный интерес для сохранения транслитерированного варианта, а также внутренний идентификатор пользователя в социальной сети. Данный набор сведений не является достаточным даже для самого элементарного анализа.

```
def scrape(facebook_id)
  authorization if !@logged_in
  @body = begin
```

```

        @agent.get("facebook_id").body
    rescue
        puts "Problem with getting fb profile".red
        nil
    end
end

def current_location
    begin
        @body.scan(/Lives in .*? href=\"(.*?)\"?ref=br_rs/).
            flatten.
            first.
            scan(/https:\\\\www.facebook.com\\/pages\\/ (.*?)\\/)/.
            flatten.
            first
    rescue
        nil
    end
end

```

Из приведенного выше фрагмента кода заметно, что алгоритмы работы с социальной сетью Facebook также требуют использования регулярных выражений в значительных количествах. Успешное завершение работы, в свою очередь, не гарантируется для кода в подобном стиле, что приводит к необходимости использовать большое количество блоков обработки исключительных ситуаций.

Пользователь Facebook при посещении профиля любого другого пользователя имеет возможность получить данные о местоположении последнего с точностью до города. Что указывает на то, что данные о местоположении доступны через веб-интерфейс. Однако использование для их получения библиотеки Koala не представляется возможным.

Как и в случае с получением данных из профессиональной социальной сети LinkedIn, нарушение политики в области данных вызывает активное противодействие со стороны компании Facebook [21].

Несложно заметить, что получение страницы профиля пользователя происходит по ранее сохраненному идентификатору, возвращенному методом из библиотеки Koala. И этот способ действительно работает первые несколько сотен раз. Дело в том, что для доступа к профилям в веб-интерфейсе социальной сети используются иные идентификаторы, не совпадающие по назначению с указанными выше и полученными через Koala. Повторяющиеся с

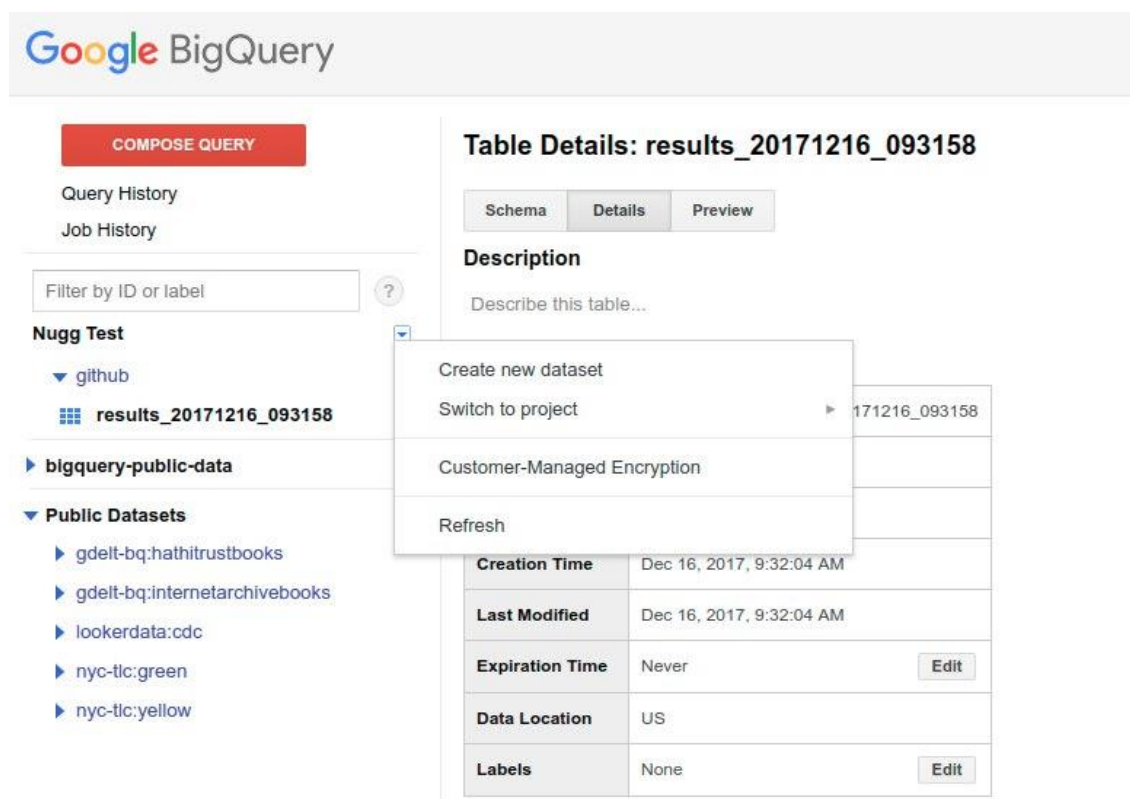


высокой частотой запросы специфического вида позволяют Facebook отслеживать нарушение политики использования данных, что приводит к отключению возможности получения профиля пользователя по идентификатору социальной сети. Любые другие возможности пользователя не блокируются и не подвергаются каким-либо изменениям. Во время тестовых запусков данного программного модуля были получены около 300 адресов с одного аккаунта социальной сети и около 100 с другого, заполненного меньшим количеством информации. Таким образом, несмотря на доказанную работоспособность данной комбинации API и синтаксического анализа, следует озаботиться поиском иных решений в случае реализации системы, задачей которой будет являться получения данных Facebook в больших масштабах. При этом, возможно, следует использовать инструменты, автоматизирующие действия пользователя в веб-браузере, так как это позволит повысить объем выполненной без блокировок работы. Хорошим способом реализовать данный функционал является библиотека Watir, доступная для использования в программном коде на языке Ruby [22].

### **3.5 Подготовка данных для коллаборативной фильтрации**

Для успешного построения рекомендательной системы по методу коллаборативной фильтрации, основанной на соседстве, необходимым этапом является подготовка данных. Для имеющихся учетных записей пользователей в системе Github необходимо получить интересующие их репозитории, сделать это можно с помощью предоставляемого API и библиотеки Octokit для языка Ruby. Информацию следует сохранить в базе данных. Вторым этапом является получение обучающей выборки, состоящей из данных о предпочтениях пользователей Github за определенный период времени. Данную задачу также возможно решить с помощью API, что и было сделано на начальном этапе работы. Однако дальнейшие исследования показали, что гораздо более эффективным вариантом является использование открытых наборов данных сервиса Google BigQuery, представляющих из себя запись последовательности системных событий сервиса Github [23]. Данные представлены в открытом виде, бесплатны для использования и имеют достаточный для большинства задач объем. Следует, однако, заметить, что работа с месячным отрезком событий, которого, как показывает практика, достаточно для построения адекватных рекомендаций, сопряжена с определенными трудностями, так как данные такого объема Google BigQuery уже не позволяет выгрузить напрямую в форматах csv или json. Таким образом, необходимо создать проект в Google Cloud Platform, добавить информацию о биллинге (в данном случае – кредитную карту), создать Google Cloud Storage для временного хранения

собранных данных, создать Dataset для использования в текущем проекте. На рисунке 3.2 показан интерфейс сервиса Google BigQuery в части, отвечающей за создание наборов данных.



**Рисунок 3.2 – Интерфейс сервиса Google BigQuery**

Для анализа представляют интерес пары пользователь – репозиторий, в то время как в проекте Google BigQuery доступны полные логи событий Github за определенные периоды. Для получения данных в необходимом виде необходимо составление запроса на языке SQL.

```
WITH stars AS (
  SELECT actor.login AS user, repo.name AS repo
  FROM githubarchive.month.201711
  WHERE type="WatchEvent"
),
repositories_stars AS (
  SELECT repo, COUNT(*) as c FROM stars GROUP BY repo
  ORDER BY c DESC
  LIMIT 1000
),
users_stars AS (
```

```

SELECT user, COUNT(*) as c FROM stars
WHERE repo IN (SELECT repo FROM repositories_stars)
GROUP BY user HAVING c > 10 AND c < 100
LIMIT 10000
)
SELECT user, repo FROM stars
WHERE repo IN (SELECT repo FROM repositories_stars)
AND user IN (SELECT user FROM users_stars)

```

Таким образом, в представленном запросе анализируется архив событий сервиса Github за ноябрь 2017 года для 1000 наиболее популярных репозиторийев, имеющих значимое число заинтересованных пользователей.

После запуска данного запроса на Google BigQuery был получен файл размером 3 Мб, содержащий 130000 пар пользователь – оцененный репозиторий.

### 3.6 Реализация программной части рекомендательной системы

В ходе исследования было выяснено, что наиболее эффективным для реализации задач, связанных с построением рекомендательных систем, является язык программирования Python, вследствие наличия большого числа библиотек машинного обучения и эффективных интерфейсов с языками низшего уровня, позволяющих реализовать ускорение алгоритмов на GPU. Однако остальная часть информационной системы анализа социальных сетей выполнена на языке Ruby, который более подходит для задач работы с API и визуализации результатов. Таким образом, необходимо обеспечить интерфейс между данными двумя языками в целях возможности построения рекомендательной системы средствами библиотек языка Python и визуализации результатов в языке Ruby.

Для решения данной задачи было опробовано два подхода. Первый, вызов кода языка Python непосредственно из языка Ruby с помощью библиотеки rucall, показал свою неэффективность для алгоритмов, работающих с большим количеством внешних библиотек. Запуск части необходимых методов завершался ошибкой времени выполнения. Второй подход, использованный в результате, состоит в вызове из программы на языке Ruby подпрограммы, написанной на языке Python с помощью встроенных возможностей языка.

Скрипт рекомендательной системы был разделен на две части с целью ускорения работы: обучение модели с сохранением результата в отдельный файл и запуск алгоритма построения рекомендаций на готовой модели,

загруженной из файла. Ниже представлена часть программного кода части, отвечающей за построение рекомендаций по готовой модели.

```
data = pd.read_csv("~/github_stars_november.csv", sep=',',
header=1, names=["user", "repo"])
data['user'] = data['user'].astype("category")
data['repo'] = data['repo'].astype("category")

confidence = 40
model = joblib.load('lib/model.pkl')

repos = dict(enumerate(data['repo'].cat.categories))
repo_ids = {r: i for i, r in repos.iteritems()}

def user_items(u_stars):
    star_ids = [repo_ids[s] for s in u_stars if s in
repo_ids]
    data = [confidence for _ in star_ids]
    rows = [0 for _ in star_ids]
    shape = (1, model.item_factors.shape[0])
    return coo_matrix((data, (rows, star_ids)),
shape=shape).tocsr()

user = user_items(args.stars)

def recommend(user_items):
    recs = model.recommend(userid=0, user_items=user_items,
recalculate_user=True)
    return [(repos[r], s) for r, s in recs]

print(json.dumps(recommend(user)))
```

Таким образом, построена рекомендательная система на основании метода коллаборативной фильтрации, основанного на соседстве, с мерой схожести ALS. Результат работы подпрограммы представляет собой набор список репозитория, соответствующих интересам определенного пользователя, с дополнением в виде процента схожести. Рекомендации, представленные в интерфейсе информационной системы, представлены на рисунке 3.3.

Repository	Score
frappe/charts	84.0%
i0natan/nodebestpractices	83.5%
moment/luxon	74.0%
Bogdan-Lyashenko/js-code-to-svg-flowchart	70.7%
rhymeai/papercss	53.3%
toddmotto/public-apis	48.7%
vuematerial/vue-material	47.6%
vasanthk/react-bits	47.5%
shieldfy/API-Security-Checklist	45.4%
airbnb/javascript	44.2%

**Рисунок 3.3 – Интерфейс рекомендательной системы**

### 3.7 Визуализация данных

Для любого типа задач, связанных со сбором и исследованием данных, одним из важнейших этапов является визуализация полученных результатов. Визуализация используется, прежде всего, с целью облегчения понимания результатов анализа человеком, однако разделяется на два заметно отличающихся вида: презентационную и исследовательскую. Презентационная визуализация данных используется для привлечения внимания неподготовленной аудитории, изложения доступным способом полученных результатов и промежуточных этапов анализа. Исследовательская визуализация данных ставит своей главной целью облегчение анализа и обработки данных, нахождение скрытых закономерностей и особенностей в них. Также стоит отметить гибридную форму, при которой большое внимание точности передаваемой информации, однако исследователь получает возможность влиять на представление данных с помощью интерактивных элементов с целью дальнейшего улучшения результатов и изучения скрытых закономерностей, труднодоступных для машинного анализа.

Для визуализации данных характерно наличие определенного списка требований, которыми должен обладать результат работы с целью улучшения качества восприятия. Прежде всего, это краткость, способность отобразить данные в максимально лаконичном виде, доступном для понимания и лишенном излишних деталей, отвлекающих внимание исследователя. Относительность и близость, позволяющие наглядно демонстрировать обнаруженные кластеры и иные отличительные особенности группировки данных, а также их схожесть и различие. Концентрация и контекст, представляющие из себя наличие возможности интерактивного взаимодействия пользователя с выбранными объектами изучения с целью определения его связей и иными объектами и положения в общей структуре данных.

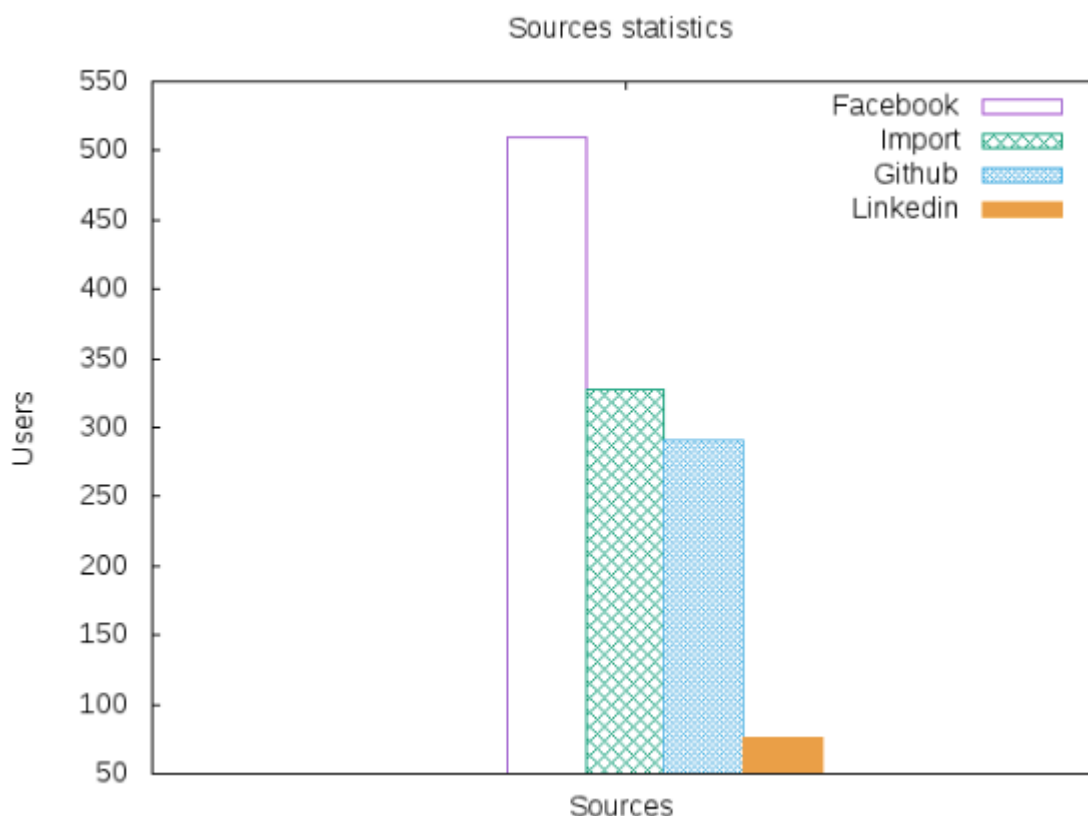
Масштабируемость – способность системы с одинаковым успехом работать с данными различного объема и перемещаться между крупномасштабным представлением результатов и более мелкомасштабными планами, обеспечивающими лучшее восприятие контекста.

Зачастую при проектировании подсистемы визуализации данных данным требованиям уделяется недостаточное внимание, что приводит к ошибочной или неправильной интерпретации результатов исследования. Тем не менее, в процессе развития прикладных инструментов анализа данных и средств представления информации в сети Интернет, появилось достаточно значительное количество законченных библиотек, позволяющих исследователю представить результаты работы в максимально доступной для понимания форме, при этом, не затрачивая на эту задачу излишнего количества времени.

Особенность реализации информационной системы анализа социальных сетей, связанная с реализацией всего программного комплекса с использованием операционной системы Linux, накладывает определенные ограничения на круг доступных для использования инструментов. Тем не менее, следует заметить, что значительная часть ученых и исследователей, так или иначе использующих визуализацию данных в своих работах, работают именно в данном пользовательском окружении. Соответственно, значительная часть рабочих инструментов доступна под систему Linux либо реализована кроссплатформенно.

Среди таких библиотек с кроссплатформенной реализацией следует особо отметить библиотеку Gnuplot, позволяющую производить построение значительного размера подмножества графиков и диаграмм, используемых в задачах анализа данных. В настоящей работе библиотека Gnuplot была использована в составе модуля генерации отчетов с помощью библиотеки `gnuplotrb`, обеспечивающей слой совместимости для вызова необходимых методов визуализации непосредственно из кода на языке Ruby. Преимущественно, используются достаточно простые и наиболее легко доступные для восприятия типы графиков, как, например, гистограммы. На рисунке 3.4 показан пример гистограммы, построенной с помощью библиотеки Gnuplot для анализируемого в данной работе набора данных.

Среди библиотек, основной задачей которых является визуализация данных, стоит также упомянуть библиотеку `Daru`, позволяющую строить интерактивные графики и гистограммы в системе интерактивного выполнения команд Jupyter. Наиболее необходимыми видами визуализации для настоящей информационной системы являются `DataFrame` – представление информации в виде таблиц и `Bar` – обыкновенные гистограммы, что является особенностью собранных данных, плохо пригодных для отображения в виде линейных графиков и кластерном представлении.

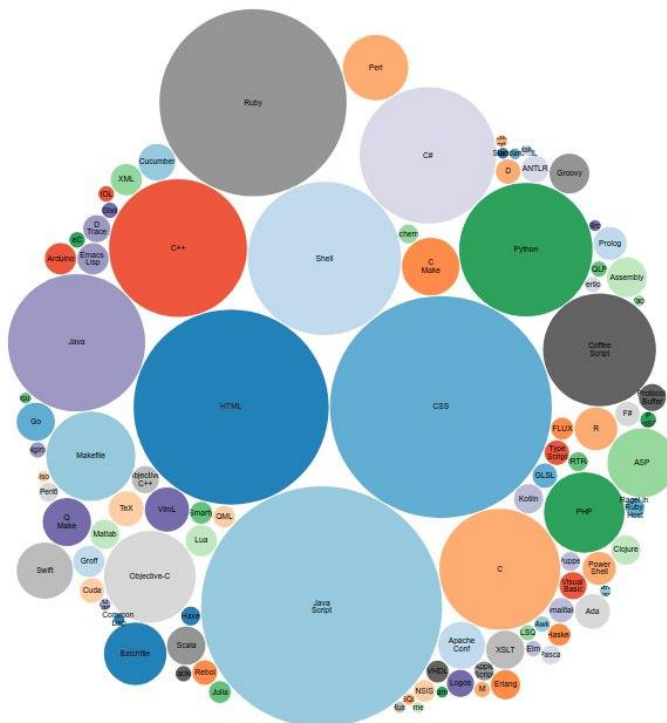


**Рисунок 3.4 – Гистограмма, построенная средствами библиотеки Gnuplot**

Наиболее широко используемой для визуализации данных библиотекой является d3.js – библиотека на языке JavaScript для встраивания отображения данных в веб-страницы. Большинство графиков, при этом, являются интерактивными и доступны в виде примеров на официальном сайте библиотеки. Для отображения анализируемых данных ключевым является принцип относительности, и вследствие этого для отображения используемых языков программирования, пользовательских навыков из социальной сети LinkedIn и стран проживания используется тип графиков Bubble – каждый объект представляется в виде круга с размером, соответствующим частоте использования. На рисунке 3.5 представлен результат построения данного типа графиков на примере данных об используемых в репозиториях системы совместной работы Github языках программирования.

Наиболее интересными среди прочих способов отображения данных представляют интерактивные, позволяющие в процессе взаимодействия изучать связи между различными изучаемыми объектами. В составе примеров использования библиотеки d3.js присутствует тип графиков Chord, пригодный для отображения связей между языками программирования. Обычно, в одном проекте используется больше одного языка программирования и совокупность используемых языков не является случайной, но несет определенную

информацию о сфере использования языков в совокупности и каждого в частности. Пример подобного графика, изображенный на рисунке 3.6, не может отобразить интерактивности, однако позволяет оценить важность представления связей между объектами.



**Рисунок 3.5 – График типа Bubble для данных о языках программирования**

Для интерактивной визуализации результатов анализа данных используется библиотека Daru, позволяющая являющаяся одним из немногих доступных для языка Ruby средств анализа данных [24]. Традиционно язык Python является более приспособленным для использования в данном направлении.

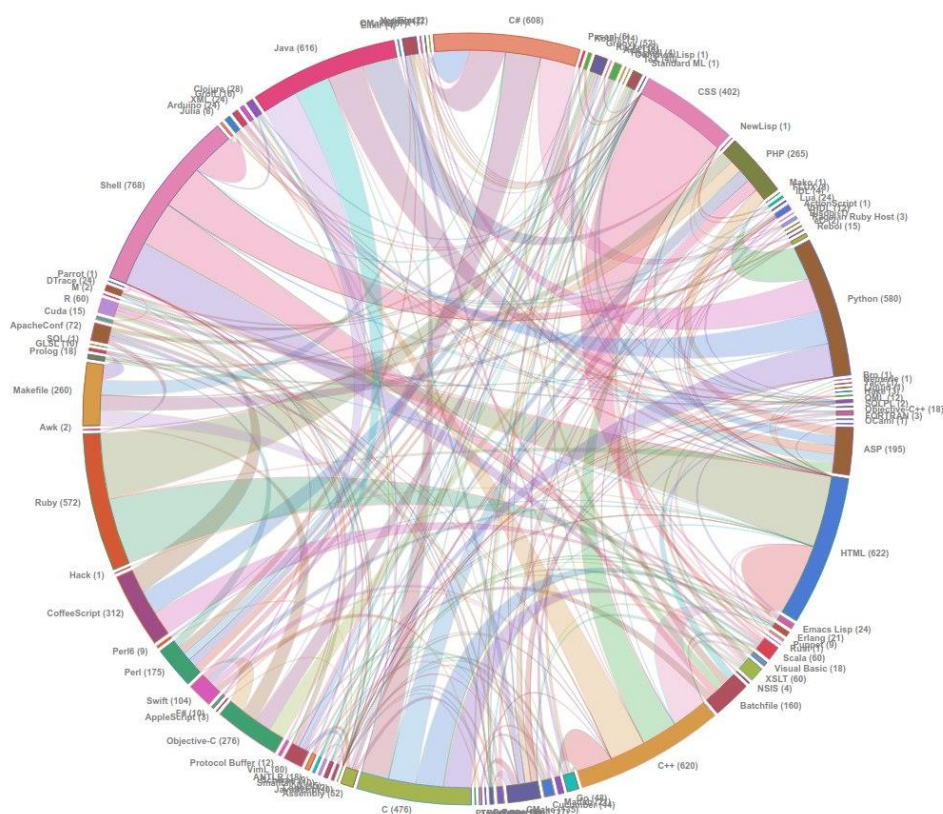
Для контейнеров DataFrame библиотека Daru позволяет использовать основные методы интерактивной работы с наборами данных: выделение векторов данных, манипуляции значениями таблицы, вычисление средних значений и проведение различного вида интерполяций результатов.

Таблица, как представление данных в форматированном виде, является в достаточной степени наглядным решением для среды интерактивной работы. Однако для формирования отчетов и визуального определения зависимостей между различными наборами данных гораздо лучше подходят графики, представленные в виде изображений и формируемые динамически на основании изменений контейнеров DataFrame.

Контейнеры `Daru::DataFrame` используются в качестве источников данных для построения графиков. Средства для построения графиков, а именно



класс Plot, предоставляются библиотекой Gnuplotrb, входящей в пакет библиотек для организации научных вычислений на языке Ruby. Библиотека Gnuplotrb, в свою очередь, действует с помощью системной библиотеки Gnuplot, входящей в базовую поставку операционных систем семейства Linux [25].



**Рисунок 3.6 – Отображение связей между языками программирования**

Таким образом, комбинирование средств, предоставляемых библиотеками Daru и Gnuplotrb, а также средства интерактивного взаимодействия и вычислений Jupyter позволило сформировать удобную среду, пригодную для выполнения различного рода операций над данными, полученными из профессиональной социальной сети LinkedIn, социальной сети Facebook и сервиса совместной работы Github.

Следует заметить, что система Jupyter не предназначена для использования в совокупности с языком программирования Ruby [26]. Для запуска среды в контексте настоящего программного модуля и получения доступа к его методам, классам и импортированным данным, использовалась библиотека IRuby, предоставляющая ядро, поддерживающее язык Ruby для среды Jupyter, работающей по умолчанию исключительно с языком программирования Python. С программным кодом, необходимым для

полнофункциональной работы указанной связки библиотек, можно ознакомиться в приложении Б к диссертации.

### 3.8 Описание модели данных

Данные, полученные из социальных сетей необходимо сохранить в удобном для обработки виде, и лучше всего для этого подходят реляционные базы данных. На этапе выбора технологий было определено, что наиболее подходящей базой для использования является PostgreSQL. Описание структуры базы данных является ключевым моментом для понимания работы программного модуля в целом и позволяет получить представление о данных, которые в дальнейшем подвергнутся обработке и анализу средствами разрабатываемого модуля. Следует отметить, что в большинство таблиц включены поля `created_at` и `updated_at`, которые соответствуют времени создания и изменения текущей записи в таблице. Также данные поля необходимы для корректного функционирования ActiveRecord, но следует заметить, что для некоторых таблиц их создание является совершенно излишним. Рассмотрим детально структуру ключевых таблиц и связи между ними, приведенную на рисунке 3.7.

#### 3.8.1 Таблица users

Данная таблица является основной. Таблицы, содержащие импортированные данные, такие как `imported_data`, `facebook_data`, `linkedin_data` и `github_data` связаны через внешние ключи с таблицей `users`, что позволяет использовать ее для упрощения запросов.

Поля:

- `id` – первичный ключ;
- `name` – транслитерированное имя пользователя, полученное из связанной таблицы;
- `facebook` – признак наличия данных, импортированных из социальной сети Facebook;
- `linkedin` – признак наличия данных, полученных из профессиональной сети LinkedIn;
- `linkedin_scraped` – признак завершения импорта из LinkedIn;
- `github` – признак наличия данных, собранных через API платформы для совместной работы Github;
- `github_scraped` – признак завершения импорта из Github;
- `import` – признак импорта из csv файлов с данными о выпускниках.

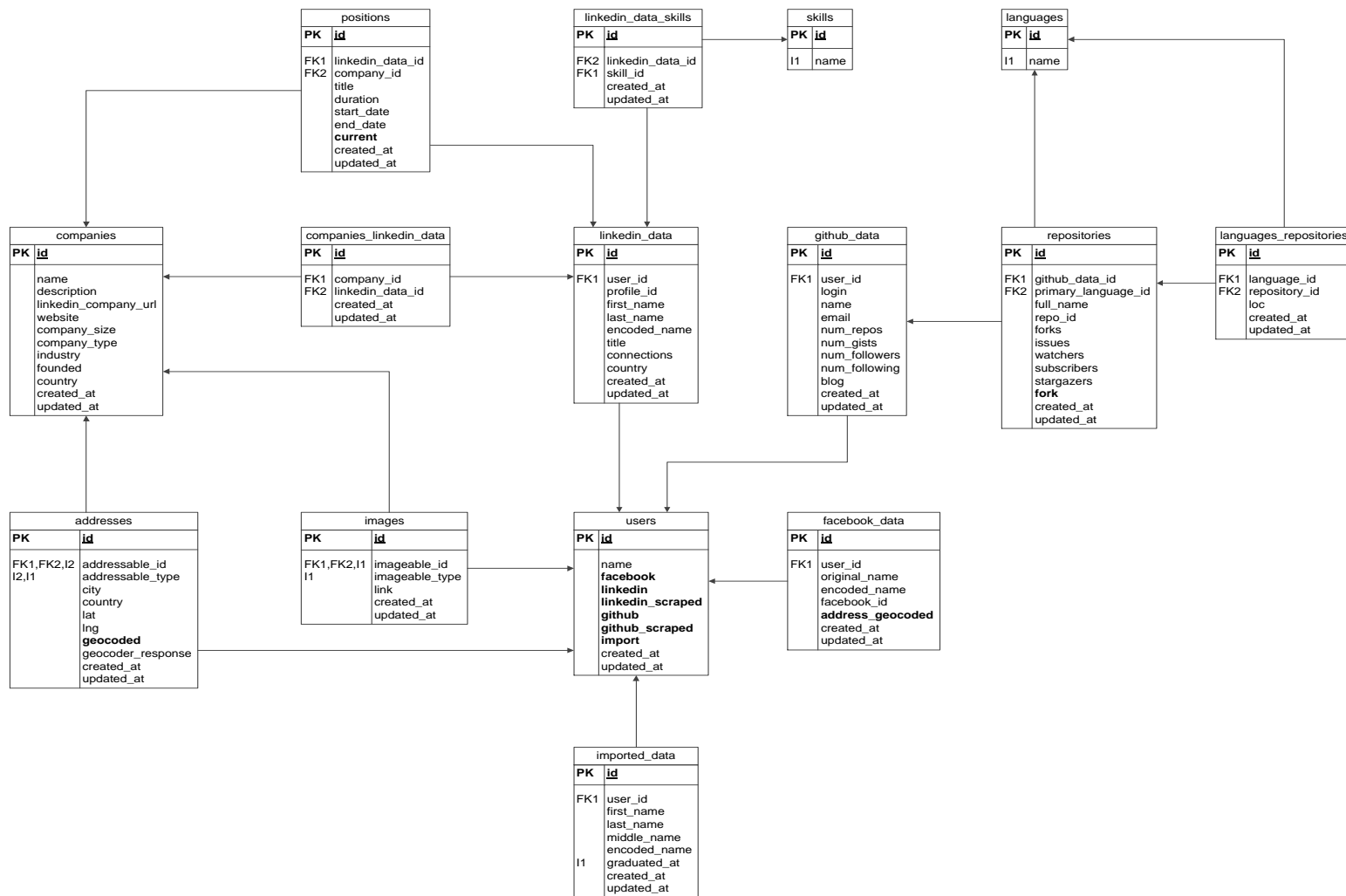


Рисунок 3.7 – Модель данных

Следует заметить, что поля `linkedin_scraped` и `github_scraped` не аналогичны по назначению другим признакам и используются для проверки факта осуществления попытки импорта из определенного источника, что позволяет не осуществлять дважды импорт отсутствующих в источнике пользователей. Все поля признаков являются обязательными.

### 3.8.2 Таблица `imported_data`

Данная таблица предназначена для хранения информации о студентах определенного года выпуска, импортированных в систему с помощью загрузки файла формата `csv`. Используется для дальнейшего поиска импортированных студентов в профессиональных социальных сетях.

Поля:

- `id` – первичный ключ;
- `user_id` – внешний ключ, связь с главной таблицей пользователей;
- `first_name` – имя студента;
- `last_name` – фамилия студента;
- `middle_name` – отчество студента;
- `encoded_name` – транслитерированное полное имя;
- `graduated_at` – дата выпуска в формате даты.

Содержит индекс по полю `graduated_at` для ускорения поиска студентов по году выпуска, что является часто встречаемой задачей при анализе данных.

### 3.8.3 Таблица `facebook_data`

Данная таблица служит для хранения данных, импортированных из социальной сети Facebook.

Поля:

- `id` – первичный ключ;
- `user_id` – внешний ключ, связь с главной таблицей пользователей;
- `original_name` – имя пользователя, указанное на странице в социальной сети;
- `encoded_name` – транслитерированное имя;
- `facebook_id` – идентификатор профиля в сети Facebook;
- `address_geocoded` – признак выполнения обработки адреса, полученного со страницы пользователя, обязательное поле.

### 3.8.4 Таблица `github_data`

Данная таблица служит для хранения данных, импортированных из сервиса совместной работы Github. Является одним из основных источников

данных для анализа, так как содержит значительное количество информации о профессиональной деятельности пользователя.

Поля:

- `id` – первичный ключ;
- `user_id` – внешний ключ, связь с главной таблицей пользователей;
- `login` – идентификатор пользователя на Github;
- `name` – имя, указанное пользователем в профиле;
- `email` – адрес электронной почты, пригодный для рассылки спама;
- `num_repos` – число репозиторий, принадлежащих пользователю (включая приватные);
- `num_gists` – число файлов, сохраненных в специальном сервисе Github Gists для хранения отдельных файлов, не входящих в проект;
- `num_followers` – число пользователей, подписанных на обновления данного профиля;
- `num_following` – число подписок на обновления профилей других пользователей;
- `blog` – адрес блога пользователя.

### 3.8.5 Таблица `linkedin_data`

Данная таблица служит для хранения данных о пользователе, импортированном из профессиональной социальной сети LinkedIn.

Поля:

- `id` – первичный ключ;
- `user_id` – внешний ключ, связь с главной таблицей пользователей;
- `profile_id` – идентификатор пользователя в социальной сети LinkedIn, является также ссылкой на его публично доступный профиль;
- `first_name` – имя пользователя;
- `last_name` – фамилия пользователя;
- `encoded_name` – транслитерированное полное имя;
- `title` – текущий статус пользователя, для LinkedIn обычно состоит из названия текущей позиции и компании-работодателя;
- `country` – страна проживания в настоящее время.

Поле `country` в дальнейшем используется для создания адреса. Следует, однако, заметить, что пользователи в LinkedIn редко предоставляют данные о стране проживания, поле используется лишь эпизодически.

### 3.8.6 Таблица `skills`

Данная таблица служит для хранения навыков, присутствующих в системе LinkedIn и используемых пользователями для заявления о наличии экспертизы в определенной сфере.

Поля:

- id – первичный ключ;
- name – название навыка.

Таблица `skills` является первой из двух таблиц, не имеющих полей `created_at` и `updated_at` в связи с особенностью хранимых данных. Для поля `name` создан индекс, облегчающий и ускоряющий поиск навыка по названию, что является наиболее частой, по большому счету, единственной операцией с данной таблицей.

### 3.8.7 Таблица `linkedin_data_skills`

Связующая таблица между навыками и данными из LinkedIn. Является необходимым элементом для работы некоторых статистических методов.

Поля:

- id – первичный ключ;
- `linkedin_data_id` – внешний ключ на таблицу, содержащую данные импорта из LinkedIn;
- `skill_id` – внешний ключ на таблицу, содержащую навыки из системы LinkedIn.

### 3.8.8 Таблица `repositories`

Данная таблица содержит данные, относящиеся к репозиторию пользователя из системы совместной работы Github. Используется для выполнения аналитики по популярности языков программирования и оценки активности участия пользователя в движении Open Source.

Поля:

- id – первичный ключ;
- `github_data_id` – внешний ключ на таблицу, содержащую данные импорта пользователя Github, владельца репозитория;
- `primary_language_id` – внешний ключ на таблицу языков программирования, основной язык, используемый в данном репозитории;
- `full_name` – ссылка на репозиторий, содержит в себе имя пользователя и название проекта;
- `repo_id` – идентификатор репозитория на Github;
- `forks` – число форков репозитория;
- `issues` – число заявок о проблемах и предложениях;
- `watchers` – число пользователей, подписанных на обновления данного репозитория;
- `subscribers` – число подписчиков репозитория на Github;
- `stargazers` – число пользователей, отметивших репозиторий. Одна из наиболее интересных величин для анализа;

– `fork` – признак того, что репозиторий является форком другого репозитория. Обычно форки создаются для участи в разработке свободного ПО.

### 3.8.9 Таблица `languages`

Данная таблица служит для хранения названий языков программирования, используемых в проектах, размещенных на Github. Используется аналогично таблице `skills` и имеет сходную структуру.

Поля:

- `id` – первичный ключ;
- `name` – название языка программирования.

Таблица `languages` является второй из двух таблиц, не имеющих полей `created_at` и `updated_at` в связи с особенностью хранимых данных. Для поля `name` создан индекс, облегчающий и ускоряющий поиск навыка по названию, что является наиболее частой, по большому счету, единственной операцией с данной таблицей.

### 3.8.10 Таблица `languages_repositories`

Связующая таблица между языками программирования и репозиториями из системы совместной работы над проектами Github. Содержит, в дополнение к внешним ключам, число строк конкретного языка в конкретном репозитории, что является неплохим источником данных для анализа в будущем.

Поля:

- `id` – первичный ключ;
- `language_id` – внешний ключ на таблицу с языками программирования;
- `repository_id` – внешний ключ на таблицу, содержащую репозиторий, полученный через API Github;
- `loc` – число строк с данным языком в репозитории. Имеет формат, позволяющий хранить в базе PostgreSQL большие числа.

### 3.8.11 Таблица `companies`

Данная таблица служит для хранения данных о компаниях, в которых работают на настоящий момент или работали в прошлом пользователи, обнаруженные программным модулем в системе LinkedIn. Служит источником аналитики по распределению компаний на мировой карте.

Поля:

- `id` – первичный ключ;
  - `name` – название компании;
  - `description` – публично доступное описание компании с LinkedIn.
- Имеет тип `text`, хорошо подходящий для хранения больших объемов текста;

- `linkedin_company_url` – ссылка на страницу компании на LinkedIn;
- `website` – ссылка на сайт компании;
- `company_size` – число сотрудников в компании. Несмотря на название, подразумевающее численный тип, поле является строковым, что связано с особенностями возвращаемого LinkedIn интервального значения;
- `company_type` – форма собственности компании;
- `industry` – сфера бизнеса компании;
- `founded` – дата основания компании, с точностью до года;
- `country` – страна, в юрисдикции которой находится компания в настоящий момент.

### 3.8.12 Таблица `companies_linkedin_data`

Связующая таблица между компаниями и данными пользователя LinkedIn. По своей сути отображает отношения между работником и работодателем. Что интересно, данная таблица не является единственной связующей таблицей этой связки, похожую функцию может выполнять таблица `positions`.

Поля:

- `id` – первичный ключ;
- `company_id` – внешний ключ на таблицу с компаниями;
- `linkedin_data_id` – внешний ключ на таблицу, содержащую данные, импортированные из LinkedIn.

### 3.8.13 Таблица `positions`

Данная таблица, помимо своей основной функции хранения данных о рабочей позиции, является также второй связующей таблицей между `companies` и `linkedin_data`.

Поля:

- `id` – первичный ключ;
- `company_id` – внешний ключ на таблицу с компаниями;
- `linkedin_data_id` – внешний ключ на таблицу, содержащую данные, импортированные из LinkedIn;
- `title` – наименование рабочей позиции;
- `duration` – длительность работы на текущем месте в виде строки с описанием на естественном языке;
- `start_date` – дата начала работы в компании;
- `end_date` – дата конца работы в компании;
- `current` – признак того, что пользователь работает на данной позиции в настоящий момент.



### 3.8.14 Таблица `addresses`

Данная таблица хранит адреса в расширенном формате. В то время, как из некоторых других таблиц возможно получить положение с точности до страны, таблица адресов содержит адрес с точностью до города с указанием географических координат. Связана с пользователями и компаниями через полиморфную связь.

Поля:

- `id` – первичный ключ;
- `addressable_id` – внешний ключ полиморфной связи;
- `addressable_type` – тип полиморфной связи;
- `city` – город;
- `country` – страна;
- `lat` – географическая широта. Используется формат с большой точностью;
- `lng` – географическая долгота. Так же, как и с случае широты, используется числовой формат с большой точностью;
- `geocoded` – признак завершения геокодинга, выполняемого с целью получения широты и долготы на основании известного текстового адреса;
- `geocoder_response` – единственное в базе данных поле типа `json`. Хранит ответ сервиса Google Geo с координатами места.

Для таблицы адресов созданы два индекса. Первый, индекс по полю `addressable_type` позволяет быстро выделять адреса, принадлежащие определенным категориям объектов, например, адреса компаний. Второй индекс по полям `addressable_id` и `addressable_type` считается необходимым для корректной работы полиморфной связи.

### 3.8.15 Таблица `images`

Вторая таблица с полиморфной связью. Служит для хранения ссылок на изображения-аватары пользователя из LinkedIn и Github, а также логотипов компаний.

Поля:

- `id` – первичный ключ;
- `imageable_id` – внешний ключ полиморфной связи;
- `imageable_type` – тип полиморфной связи;
- `link` – ссылка на картинку.

Данная таблица с полиморфной связью содержит лишь один индекс, по полям `imageable_id` и `imageable_type`, который считается необходимым для корректной работы полиморфной связи. Во введении индекса по типу полиморфной связи не возникло необходимости, в связи с отсутствием возможности получения из изображений информации, пригодной для анализа.

### 3.9 Описание структуры и взаимодействия между классами

Рассмотренная в предыдущих пунктах модель данных позволяет оценить структуру и взаимосвязи данных в информационной системе анализа социальных сетей. Однако для полного описания результата разработки необходимо также рассмотреть структуру основных классов проекта и их взаимодействие, а также перечень основных методов и полей.

#### 3.9.1 Класс Geo

Класс Geo является базовым классом для AddressableGeo и FacebookGeo и составляет вместе с ними основу модуля геоданных. Данный класс предназначен для преобразования адреса, представленного в виде строки, содержащей страну и, возможно, город нахождения пользователя, в объект базы данных, содержащий в себе точные географические координаты соответствующего места. Для выполнения данной задачи используется сервис геокодинга, предоставляемый Google. Данный класс содержит только приватные методы, что объясняется его использованием в качестве базового.

Методы класса:

- geocode(address) – преобразование строки с адресом в json объект, содержащий точные названия страны и города, а также их географические координаты. В данном методе используется сторонняя библиотека Geocoder, позволяющая преобразовывать адреса с использованием публичных сервисов группы компаний Alphabet;

- save(resp, id, type) – сохранение полученного на предыдущем шаге json в таблицу addresses базы данных. Параметры id и type представляют собой пару значений, необходимых для создания полиморфной связи между адресом с одной стороны и компанией либо пользователем с другой. Метод содержит отладочный вывод, предназначенный для визуальной индикации работы в интерактивном режиме;

- update(resp, address) – в отличие от метода save, данный метод предназначен не для сохранения нового адреса в базу данных, а для изменения уже существующего. Предназначена эта возможность для случаев, когда некоторая часть адреса, например, город, уже была сохранена, однако необходимо получить координаты для дальнейшего импорта. Следует заметить, что адрес, прошедший процедуру геокодинга, помечается флагом geocoded.

#### 3.9.2 Класс WebScraper

Является базовым классом модуля синтаксических анализаторов. Предназначен для проведения первичной инициализации анализатора веб-

страниц Mechanize, который предоставляется сторонней библиотекой и позволяет получить объектное отображение HTML кода страницы.

Поля класса:

- agent – объект, содержащий анализатор HTML страниц Mechanize.

Методы класса:

- initialize() – конструктор, производит инициализацию поля agent и первичную настройку параметров библиотеки Mechanize.

### **3.9.3 Класс Export**

Данный класс реализует функции модуля экспорта, предоставляя данные, подвергнутые анализу, в необходимом для использования в модуле отчетов и интерактивном режиме виде.

Методы класса:

- addresses\_for\_map(countries) – метод класса, экспортирующий статистику по странам в формат csv для дальнейшего использования при отображении мировой карты;
- encode(image) – приватный метод, переводит изображение в закодированный формат Base64, что позволяет встроить строку, содержащую закодированное изображение в тег img HTML страницы при формировании отчета модулем отчетов;
- sources() – предоставляет статистику по источникам данных в виде строки Base64;
- graduation\_date() – предоставляет статистику по датам выпуска студентов в виде строки Base64;
- countries() – предоставляет статистику по странам пользователей в виде строки Base64;
- skills() – предоставляет статистику по навыкам, собранным в системе LinkedIn, в виде строки Base64;
- languages\_in\_repos() – предоставляет статистику по языкам программирования, используемых в репозиториях сервиса Github в виде строки Base64;
- companies\_countries() – предоставляет статистику по странам происхождения организаций в виде строки Base64;
- company\_sizes() – предоставляет статистику по размерам организаций в виде строки Base64.

### **3.9.4 Класс Report**

Данный класс образует собой модуль формирования отчетов. Закодированные в Base64 изображения, полученные с помощью модуля экспорта, передаются в особом образом сформированное представление и

выводятся в файл формата pdf средствами библиотеки WickedPdf. Данная библиотека позволяет преобразовывать файлы формата HTML в формат pdf и в данном виде в большинстве промышленно разработанных приложений, имеющих в своем составе функционал генерации отчетов.

Поля класса:

- view – объект типа ActionView, содержащий шаблон, подготовленный для преобразования в формат pdf;

- save\_path – путь к сгенерированному файлу отчета.

Методы класса:

- save() – публичный метод, сохраняет файл с отчетов по заданному пути;

- rendered\_pdf() – формирует документ формата pdf из представления, сформированного методом rendered\_view;

- rendered\_view() – формирует представление ActionView и передает в него необходимые внутренние переменные.

### 3.9.5 Класс Analytics

Данный класс формирует структуры данных, необходимые для дальнейшего построения графиков и создания отчетов и представляет модуль анализа данных. Аналитика используется относительно простая, однако следует заметить, что в сочетании с полученными данными это дает интересные с точки зрения общей информации результаты. Методы, включенные в данный класс, являются методами класса и возвращают, в основной массе, объекты класса Daru::DataFrame. Daru – библиотека анализа данных на языке Ruby, используемая в настоящем проекте.

Методы класса:

- count\_in\_array(array) – метод, подсчитывающий число повторяющихся элементов в массиве и возвращающий объект типа hash с парами значение – число повторений;

- sources() – предоставляет статистику по источникам данных в виде объекта Daru::DataFrame;

- graduation\_date() – предоставляет статистику по датам выпуска студентов в виде объекта Daru::DataFrame;

- countries() – предоставляет статистику по странам пользователей в виде объекта Daru::DataFrame;

- skills() – предоставляет статистику по навыкам, собранным в системе LinkedIn, в виде объекта Daru::DataFrame;

- languages\_in\_repos() – предоставляет статистику по языкам программирования, используемых в репозиториях сервиса Github в виде объекта Daru::DataFrame;

- `companies_countries()` – предоставляет статистику по странам происхождения организаций в виде объекта `Daru::DataFrame`;
- `company_sizes()` – предоставляет статистику по размерам организаций в виде объекта `Daru::DataFrame`.

### 3.9.6 Класс Visualization

Данный класс формирует графики типа гистограмма из предоставленных объектов класса `Daru::DataFrame`. Для построения графиков используется библиотека `Gnuplotrb`, основанная на системной библиотеке `Gnuplot`. Выбор пал именно на этот вариант благодаря беспроблемной работе данной связки в системе интерактивного выполнения команд `Jupyter`.

Методы класса:

- `sources()` – предоставляет статистику по источникам данных в виде графика, образованного объектом `Plot`;
- `graduation_date()` – предоставляет статистику по датам выпуска студентов в виде графика, образованного объектом `Plot`;
- `countries()` – предоставляет статистику по странам пользователей в виде графика, образованного объектом `Plot`;
- `skills()` – предоставляет статистику по навыкам, собранным в системе `Linkedin`, в виде графика, образованного объектом `Plot`;
- `languages_in_repos()` – предоставляет статистику по языкам программирования, используемых в репозиториях сервиса `Github` в виде графика, образованного объектом `Plot`;
- `companies_countries()` – предоставляет статистику по странам происхождения организаций в виде графика, образованного объектом `Plot`;
- `company_sizes()` – предоставляет статистику по размерам организаций в виде графика, образованного объектом `Plot`.

### 3.9.7 Класс SocialAnalytics

Данный класс образует интерфейс для разрабатываемого модуля анализа данных профессиональных социальных сетей и включает в себя методы, содержащие необходимую последовательность действий для обработки данных из любого из определенных на этапе анализа предметной области источников. Все необходимые данному модулю пароли рекомендуется хранить в файле, исключенном из системы контроля версий.

Поля класса:

- `facebook_token` – токен `OmniAuth`, необходимый для получения данных через `API Facebook` с помощью библиотеки `Koala`;
- `facebook_email` – логин для веб-интерфейса `Facebook`, необходимый для работы синтаксического анализатора `FacebookWebScraper`;

- `facebook_password` – пароль для веб-интерфейса Facebook, необходимый для работы синтаксического анализатора `FacebookWebScraper`;
- `linkedin_email` – логин для веб-интерфейса LinkedIn, необходимый для работы синтаксического анализатора `LinkedinWebScraper`;
- `linkedin_password` – пароль для веб-интерфейса LinkedIn, необходимый для работы синтаксического анализатора `LinkedinWebScraper`;
- `github_login` – логин для Github, требуемый для получения данных через API;
- `github_password` – пароль для Github, требуемый для получения данных через API.

Методы класса:

- `handle_facebook()` – публичный метод, реализующий необходимую последовательность действий для получения пользователей из группы Facebook и обработки всех полученных адресов;
- `handle_linkedin()` – публичный метод, реализующий необходимую последовательность действий для получения пользователей из профессиональной социальной сети LinkedIn на основе предварительно импортированных из файлов csv выпускников и пользователей, полученных из группы на Facebook и сбора дополнительной информации о навыках и организациях;
- `handle_github()` – публичный метод, реализующий необходимую последовательность действий для получения пользователей из сервиса совместной работы Github на основе предварительно импортированных из других источников данных и сбора дополнительной информации об организациях, репозиториях и используемых языках программирования.

## **ГЛАВА 4**

### **ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

Наиболее значимой задачей в жизненном цикле любого программного обеспечения, помимо небольших программ, созданных для работы над конкретными исследовательскими задачам, является тестирование с целью выявления ошибок и определения поведения системы условиях, приближенных к реальным. Каждый этап разработки системы сопровождается написанием разнообразных тестов и каждая новая порция функционала проверяется различными методами, такими как:

- функциональное тестирование;
- тестирование производительности;
- тестирование стабильности;
- тестирование совместимости;
- тестирование интерфейса пользователя.

#### **4.1 Автоматизированное тестирование**

На сегодняшний день существует значительное количество систем промышленной проверки качества программ. Для фреймворка Ruby on Rails такой системой по умолчанию является модуль TestUnit, позволяющий производить тестирование моделей, контроллеров и представлений. Затратив минимум усилий можно выполнить три вида тестирования: модульное, функциональное и комплексное.

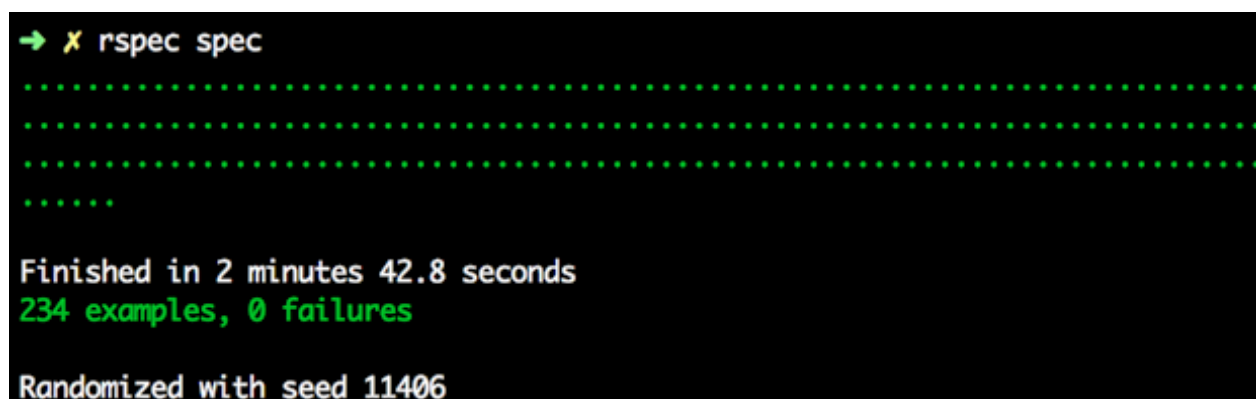
При написании тестов для различных участков программного кода приложения значительно уменьшаются риски нарушения функционирования проекта в процессе внесения изменений и улучшений. Полное покрытие кода тестами занимает от 38 до 50 процентов времени работы над проектом, но временные затраты окупаются увеличением надежности и упрощением совместной работы над критическими изменениями, так как, по мере продвижения работы над проектом, стоимость устранения дефектов ПО может и будет экспоненциально возрастать. Статический и динамический анализ программного кода позволяют обнаружить программные ошибки на ранней стадии, значительно упростить их устранение и удешевить его.

Следует, однако, заметить, что для исследовательских проектов использование автоматизированного тестирования характерно в меньшей степени, чем для коммерческих.

При написании программной части данной диссертации использовался принцип TDD (Test Driven Development), написание программного кода через тестирование. Основным принцип данной техники разработки состоит в

повторении коротких циклов написания кода и тестов, а также последующего рефакторинга получившихся алгоритмов.

Было достигнуто полное покрытие тестами моделей, контроллеров и представлений части информационной системы, ответственной за сбор данных и пользовательский интерфейс, написанной с использованием языка Ruby и фреймворка Ruby on Rails, как показано на рисунках 4.1 и 4.2.

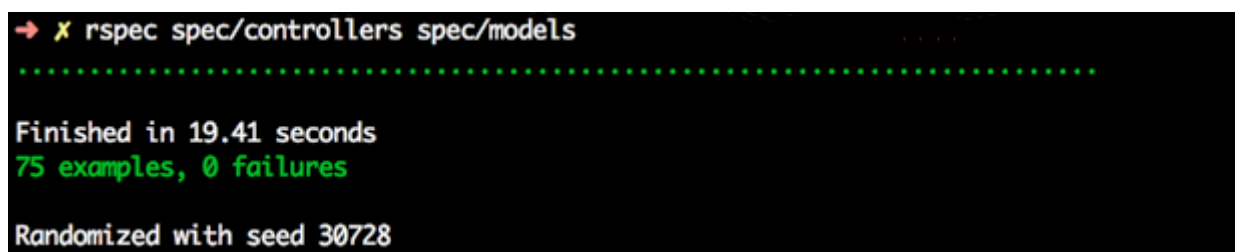


```
→ ✗ rspec spec
.....
.....
.....
.....

Finished in 2 minutes 42.8 seconds
234 examples, 0 failures

Randomized with seed 11406
```

Рисунок 4.1 – Результаты выполнения общего тестирования системы



```
→ ✗ rspec spec/controllers spec/models
.....

Finished in 19.41 seconds
75 examples, 0 failures

Randomized with seed 30728
```

Рисунок 5.2 – Результаты выполнения тестирования контроллеров

## 4.2 Ручное тестирование

Следует заметить, что алгоритмы анализа данных, реализованные в скриптах на языке Python, и части интерфейса, ответственные за визуализацию данных, являются более трудными задачами для автоматизированного тестирования. Специфика проекта и данных частей функционала позволяют использовать в их отношении ручное тестирование.

Рекомендательная система, работающая с данными о предпочтениях пользователей системы Github, формирует рекомендации по проектам для дальнейшего обучения, отображаемые вместе с оценкой степени достоверности.

С целью обучения рекомендательной системы, модель обучается на выборке, полученной из сервиса Google BigQuery. Среднее время обучения составляет около 20 секунд, однако после первоначального обучения имеется



возможность сохранить обученную модель в отдельный файл и использовать его при формировании рекомендаций. Общее время построения рекомендаций для конкретного пользователя, таким образом, не превышает десяти секунд и зависит от объема доступной о нем информации.

Рассмотрим результаты работы данной рекомендательной системы для нескольких пользователей.

Например, пользователь с логином `ilyaravlov` имеет в закладках проекты, относящиеся к дизайну веб-страниц, например, `bootstrap` и `font-awesome`, а также популярные, но устаревшие Javascript фреймворки и библиотеки, например, `Jquery`. Из сферы интересов пользователя также можно выделить SEO – поисковое продвижение, которое представлено проектом `wordpress-seo`. В рекомендациях, сформированных для данного пользователя и показанных на рисунке 4.3 можно заметить проекты тех же категорий, из области дизайна присутствуют `material-design-icons`, `Chart.js` и `animate.css`. Javascript-фреймворки представлены более актуальными на сегодняшний день `vue`, `ionic` и `angular`, что позволяет сделать вывод о корректной работе рекомендательной системы. SEO также представлено в списке рекомендаций – это проект `Semantic-UI`.

Repository	Score
<code>daneden/animate.css</code>	62.5%
<code>angular/angular.js</code>	57.0%
<code>Semantic-Org/Semantic-UI</code>	55.2%
<code>nicolas/normalize.css</code>	54.5%
<code>google/material-design-icons</code>	53.2%
<code>github/gitignore</code>	50.5%
<code>ionic-team/ionic</code>	50.0%
<code>Microsoft/vscode</code>	46.8%
<code>chartjs/Chart.js</code>	44.9%
<code>vuejs/vue</code>	44.5%

**Рисунок 4.3 – Рекомендации для пользователя `ilyaravlov`**

Пользователь `idej` интересуется машинным обучением, судя по отмеченным проектам `catboost` и `h2o`, а также модулируемыми списками библиотек и технологий для определенных языков, такими как `awesome-ruby`. Среди прочего его интересуют проекты, содержащие советы о прохождении интервью, как то `coding-interview-university`. Рекомендации, показанные на рисунке 4.4 содержат проекты `every-programmer-should-know`, `papers-we-love`, которые являются аналогами `awesome-ruby` и `coding-interview-university`, а также библиотеку машинного обучения `keras` и примеры для библиотеки `TensorFlow`. Точность работы для данного пользователя также можно оценить как удовлетворительную.

Таким образом, в целом система построения рекомендаций работает удовлетворительным образом и обладает достаточной точностью для предсказания будущих интересов пользователей.

Следует также протестировать работу интерактивных графиков подсистемы визуализации данных, реализованных на языке Javascript с помощью библиотеки d3.js.

Repository	Score
papers-we-love/papers-we-love	113.9%
mr-mig/every-programmer-should-know	101.7%
sdmg15/Best-websites-a-programmer-should-visit	98.5%
jlevy/the-art-of-command-line	87.3%
posquit0/Awesome-CV	85.3%
resume/resume.github.com	85.0%
vuejs/vue	83.2%
oxford-cs-deepnlp-2017/lectures	82.5%
fchollet/keras	79.5%
aymericdamien/TensorFlow-Examples	78.7%

**Рисунок 4.4 – Рекомендации для пользователя idej**

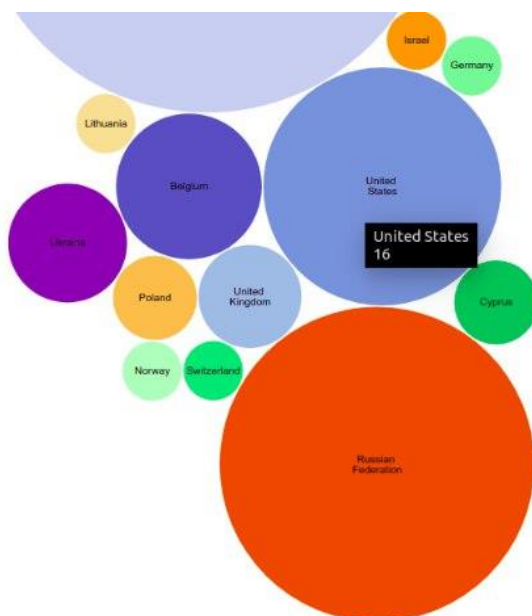
Для визуализации информации о странах используется мировая карта с отображением количества объектов в каждой территориальной единице оттенками синего цвета, как показано на рисунке 4.5, что в достаточной степени отвечает требованиям наглядности. Для каждой страны возможен вывод точного количества объектов по наведению курсора.



**Рисунок 4.5 – Распределение пользователей на мировой карте**

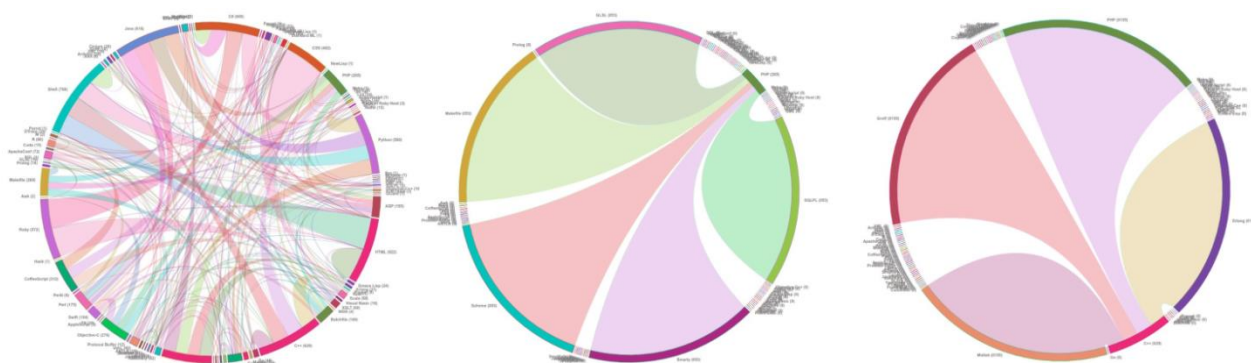
В качестве альтернативного варианта отображения информации о странах используется график типа bubble, представляющий каждую страну в виде круга

определенного размера и цвета, зависящих от количества объектов в данной территориальной единице, как показано на рисунке 4.6, что обеспечивает большую относительную наглядность и меньшую степень достоверности в плане точного географического положения.



**Рисунок 4.6 – Альтернативное представление данных о странах**

Наиболее интересным для тестирования интерактивности является график взаимосвязи языков в проектах системы Github. На рисунке 4.7 изображены различные фазы, в которых может находиться график при выборе различных языков в качестве основы для произведения наблюдений. Первая фаза представляет собой состояние по умолчанию, вторая показывает языки, используемые одновременно с PHP, а третья – одновременно с C++.



**Рисунок 4.7 – Поведение интерактивного графика взаимосвязей языков**

## ЗАКЛЮЧЕНИЕ

В рамках работы над диссертацией была спроектирована и разработана информационная система анализа социальных сетей Github, Facebook и LinkedIn, содержащая в своем составе рекомендательную систему на коллаборативной фильтрации по методу соседства с использованием алгоритма чередующихся наименьших квадратов.

Было проведено исследование данных студентов и выпускников специальности ВМСиС в вышеуказанных социальных сетях. Наиболее интересными из полученных результатов являются статистика распределения студентов и их работодателей по странам, данные о декларируемых навыках и умениях из профессиональной социальной сети LinkedIn, информация о использовании студентами различных языков программирования в процессе работы над проектами с открытым исходным кодом в системе совместной работы Github. Для студентов, имеющих аккаунты в системе Github, были сформированы рекомендации по наиболее интересным для дальнейшего изучения направлениям и проектам. С целью улучшения восприятия результатов анализа данных был разработан интерфейс информационной системы с использованием фреймворка Ruby on Rails, содержащий табличное представление основных анализируемых данных, а также графики различных видов, созданные с помощью библиотеки d3.js. Из всех графиков наиболее интересным является график связей между языками программирования, построенный исходя из собранных данных о проектах студентов в системе Github, и позволяющий путем интерактивного взаимодействия исследовать одновременное использование языков программирования.

Также для лучшего отображения результатов в информационную систему была интегрирована система Jupyter, позволяющая не только использовать уже готовые методы анализа и визуализации, но и создавать новые с минимальными затратами усилий, в том числе, и людям, не имеющим полного спектра технических навыков.

В процессе работы с внешними источниками данных и использования программных интерфейсов исследуемых социальных сетей была приобретена экспертиза в области data mining.

Имеет определенные перспективы использование полученной информации в процессе работы на учебными планами магистратуры и бакалавриата университета с целью предоставления наиболее актуальных знаний, востребованных на международном рынке труда.

## СПИСОК ЛИТЕРАТУРЫ

- [1] LinkedIn takes aim at developers with plans to lock down most of its APIs [Электронный ресурс]. – Режим доступа: <http://thenextweb.com/dd/2015/02/12/linkedin-takes-aim-developers-plans-lock-apis/#gref>.
- [2] Github is the next big social network, powered by what you do, not who you know [Электронный ресурс]. – Режим доступа: <http://www.forbes.com/sites/anthonykosner/2012/07/15/github-is-the-next-big-social-network-powered-by-what-you-do-not-who-you-know/#1a76409179fd>.
- [3] Система аналитики Brandwatch [Электронный ресурс]. – Режим доступа: <https://www.brandwatch.com/>.
- [4] Система аналитики 33Across [Электронный ресурс]. – Режим доступа: <http://www.33across.com>.
- [5] Система аналитики Hootsuite [Электронный ресурс]. – Режим доступа: <https://hootsuite.com>
- [6] Moz [Электронный ресурс]. – Режим доступа : <https://moz.com>.
- [7] Github analysis and discovery [Электронный ресурс]. – Режим доступа: <https://github.com/anvaka/gazer>
- [8] Building recommender system for Github [Электронный ресурс]. – Режим доступа: [https://medium.com/@andrey\\_lisin/building-recommender-system-for-github-a8108f0cb2bd](https://medium.com/@andrey_lisin/building-recommender-system-for-github-a8108f0cb2bd)
- [9] Язык Ruby: история становления и перспективы развития [Электронный ресурс]. – Режим доступа: <http://habrahabr.ru/post/131661>.
- [10] Ruby учебник [Электронный ресурс]. – Режим доступа: <http://ru.wikibooks.org/wiki/Ruby>.
- [11] Исходный код языка Ruby [Электронный ресурс]. – Режим доступа: <https://github.com/ruby/ruby>.
- [12] Ruby community gem host [Электронный ресурс]. – Режим доступа: <http://rubygems.org>.
- [13] Su Xiaoyuan Taghi A survey of collaborative filtering techniques / Xiaoyuan Su // Advances in Artificial Intelligence – 2009. – January. – P.4.
- [14] Recommending Github Repositories with Google BigQuery and the implicit library [Электронный ресурс]. – Режим доступа: <https://towardsdatascience.com/recommending-github-repositories-with-google-bigquery-and-the-implicit-library-ебссе666с77>
- [15] Ключкова А.А. Исследование и разработка рекомендательной системы музыкальных композиций, работающей на основе неявных

пользовательских оценок: дис. ст: 02.04.03 / А. А. Ключкова. – Санкт Петербург: 2017. – 61с.

[16] Наиболее продвинутая открытая СУБД в мире [Электронный ресурс]. – Режим доступа: <http://postgresql.ru.net/docs/overview.html>.

[17] Github API developer guide [Электронный ресурс]. – Режим доступа: <https://developer.github.com/v3>.

[18] Octokit official repository [Электронный ресурс]. – Режим доступа: <https://github.com/octokit/octokit.rb>.

[19] Scrape the public profile of linkedin page [Электронный ресурс]. – Режим доступа: <https://github.com/yatish27/linkedin-scraper>.

[20] A lightweight, flexible library for Facebook with support for OAuth authentication [Электронный ресурс]. – Режим доступа: <https://github.com/arsduo/koala>.

[21] Facebook Data Policy [Электронный ресурс]. – Режим доступа: <https://facebook.com/policy.php>.

[22] Watir: web application testing in Ruby [Электронный ресурс]. – Режим доступа: <https://watir.com/documentation>.

[23] Github Data: how to query public data sets using BigQuery [Электронный ресурс]. – Режим доступа: <https://cloud.google.com/bigquery/public-data/github>

[24] Data analysis in Ruby [Электронный ресурс]. – Режим доступа: <https://github.com/v0dro/daru>.

[25] Gnuplot homepage [Электронный ресурс]. – Режим доступа: <https://gnuplot.info>.

[26] Project Jupyter [Электронный ресурс]. – Режим доступа: <https://jupyter.org>.

### **Список публикаций соискателя**

[1–А] Белов А.В. Ключевые особенности анализа данных профессиональных социальных сетей/ А. В. Белов// Компьютерные системы и сети: материалы 53-й научной конференции аспирантов, магистрантов и студентов – Минск, 2017 – С. 17 – 18.

## ПРИЛОЖЕНИЕ А

### Исходный код некоторых алгоритмов сбора данных

```
module SocialAnalytics
  class << self
    attr_accessor :facebook_token, :facebook_email,
:facebook_password,
                  :linkedin_email, :linkedin_password,
:github_login,
                  :github_password
  end

  def self.handle_facebook(gid = nil)
    puts "Start receiving facebook group members...".green
    if FacebookData.count.zero?
      FacebookScraper.new(facebook_token,
gid).save_group_members
    end
    FacebookGeo.new(false).handle
  end

  def self.handle_linkedin
    scraper = LinkedinScraper.new
    User.where(linkedin: false, linkedin_scraped: false).each do
|user|
      scraper.get_public_member(user.name)
      user.update_columns(linkedin_scraped: true)
      sleep 1
    end
  end

  def self.handle_github
    scraper = GithubScraper.new
    User.where(github: false, github_scraped: false).each do
|user|
      scraper.search_user(user.name)
      user.update_columns(github_scraped: true)
      sleep 1
    end
  end

class GithubScraper
  def initialize
    @users = []
  end

  def search_user(name)
    logins =
Octokit.search_users(name).items.map(&:login).first(2)
```

```

    @users = logins.map{|login| Octokit.user(login)}
    save_user
end

def save_user
  return if @users.empty?

  puts "Found Github users".blue

  @users.each do |user|
    github_data = GithubData.create(
      login: user.login,
      name: user.name,
      email: user.email,
      num_repos: user.public_repos,
      num_gists: user.public_gists,
      num_followers: user.followers,
      num_following: user.following,
      blog: user.blog
    )
    if user.avatar_url.present?
      puts "Saving avatar image".yellow
      Image.create(
        imageable: github_data.user,
        link: user.avatar_url
      )
    end
    save_repositories(user.login)
  end
end

def save_repositories(login)
  puts "Saving repositories".green
  repositories =
Octokit.repositories(login).map(&:id).first(10)
  repositories.each do |repo_id|
    repo = Octokit.repository(repo_id)
    repository = Repository.create(
      github_data_id: GithubData.find_by_login(login),
      primary_language_id: Language.
        where(name: repo.language).
        first_or_create.id,
      full_name: repo.full_name,
      repo_id: repo_id,
      forks: repo.forks,
      issues: repo.issues,
      watchers: repo.watchers,
      subscribers: repo.subscribers,
      stargazers: repo.stargazers,
      fork: repo.fork
    )
    save_languages(repository)
  end
end

```



```

    end
  end

  def save_languages(repository)
    languages = Octokit.languages(repository.repo_id)
    languages.each do |lang, loc|
      language = Language.where(name: lang).first_or_create
      repository.languages << language
      lang_repo = LanguagesRepositories.where(
        language_id: language.id,
        repository_id: repository.id).first
      lang_repo.update_attributes(loc: loc)
    end
  end
end

class LinkedinScraper
  def initialize
    @profile = nil
    @public_profile = nil
    @web_scraper = LinkedinWebScraper.new
  end

  def get_public_member(name)
    @public_profile = @web_scraper.scrape(name)
    return if @public_profile.nil?
    @profile = begin
      Linkedin::Profile.new(
        "http://www.linkedin.com/in/#{@public_profile}",
        { company_details: true })
    rescue
      nil
    end
    save_public_member
  end

  def save_companies(companies, current = false)
    companies.each do |data|
      company = Company.where(name:
data[:company]).first_or_create do |c|
        c.description = data[:description]
        c.linkedin_company_url = data[:linkedin_company_url]
        c.website = data[:website]
        c.company_size = data[:company_size]
        c.company_type = data[:company_type]
        c.industry = data[:industry]
        c.founded = Preprocessing.
          foundation_date(data[:founded])
        c.country = data[:country]
      end

      if data[:company_url].present?

```

```

        Image.create(
          imageable: company,
          link: data[:company_logo]
        )
      end

      if !company.address.present? && data[:country].present?
        Address.create(
          addressable: company,
          country: data[:country]
        )
      end

      Position.create(
        linkedin_data: @linkedin_data,
        company_id: company.id,
        title: data[:title],
        duration: data[:duration],
        start_date: data[:start_date],
        end_date: data[:end_date],
        current: current
      )
    end
  end
end

def save_public_member
  return if !@profile.present?

  puts "Found Linkedin user - ".blue +
    "#{@profile.first_name} #{@profile.last_name}".green

  @linkedin_data = LinkedInData.create(
    profile_id: @public_profile,
    first_name: @profile.first_name,
    last_name: @profile.last_name,
    title: @profile.title,
    connections: @profile.number_of_connections.to_i,
    country: @profile.country
  )

  if @linkedin_data.addresses.empty? &&
    @profile.country.present?
    puts "Saving linkedin address".yellow
    Address.create(
      addressable: @linkedin_data.user,
      country: @profile.country
    )
  end

  if @linkedin_data.skills.empty?
    puts "Saving skills".green
    @profile.skills.each do |skill_name|

```

```

        skill = Skill.where(name: skill_name).first_or_create
        @linkedin_data.skills << skill
      end
    end

    puts "Saving companies".yellow
    save_companies(begin @profile.past_companies rescue []
end, false)
    save_companies(begin @profile.current_companies rescue []
end, true)
  end
end

class LinkedinWebScraper < WebScraper
  def scrape(name)
    authorization if !@logged_in
    url =
    "http://www.linkedin.com/vsearch/p?type=people&keywords=#{name.s
    plit(' ').join('+')}"
    @results =
    @agent.get(url).body.scan(/\{"person"\: \{.*?\}\}\}/)
    return if !@results[0].present?
    @profile_id = begin
      JSON.parse(@results[0]).
        to_s.scan(/profile\/view\?id=(.*?)&/).
        compact.first.first
    rescue
      nil
    end
    get_public_profile
  end

  private

  def authorization
    @agent.get('https://www.linkedin.com/')
    form = @agent.page.form_with(:method => 'POST')
    form.session_key = SocialAnalytics.linkedin_email
    form.session_password = SocialAnalytics.linkedin_password
    resp = @agent.submit(form)
    @logged_in = resp.title == "Welcome! | LinkedIn"
  end
end

```

## ПРИЛОЖЕНИЕ Б

### Интеграция программного модуля в систему Jupyter

```
namespace :jupyter do
  task :notebook do
    root = File.absolute_path(__FILE__ + "../../../..")
    env = {
      "RUBYLIB" => ($: + [ root ]).join(":")
    }
    Process.exec(env, "jupyter", "notebook")
  end
end

Dir.chdir File.dirname(File.dirname(__FILE__))
unless defined? Rails
  APP_PATH = File.expand_path('../../config/application',
__FILE__)
  require File.expand_path('../../config/boot', __FILE__)
  require APP_PATH
  Rails.application.require_environment!
end
puts "Loaded #{Rails.env} environment"
```