Санкт-Петербургский Государственный Университет
Кафедра Технологий программирования

Логачев Михаил Максимович

Выпускная квалификационная работа бакалавра

Детектирование общественно значимых новостей в потоке сообщений

Направление 01.03.02
Прикладная математика, фундаментальная информатика и программирование

Научный руководитель,
старший преподаватель
Малинина Мария Анатольевна

Санкт-Петербург
2018

Введение

На сегодняшний день интернет в общем, и социальные сети в частности стали неотъемлемой частью жизни людей. Тяжело найти человека, который ни разу за день не проверяет почту и не читает сообщения от семьи или коллег по работе. Интернет трансформировал обычную жизнь. Благодаря ему, у людей появилась возможность чуть ли не мгновенного общения с помощью аудио, видео или текстовых сообщений с собеседником из другой части света.

Огромную роль в процессе упрощения общения между людьми играют социальные сети. С их помощью можно найти родственников или одноклассников, с которыми давно утеряна связь. Или же найти друзей для общения на интересующую тему. В наше время, с помощью социальных сетей можно делать практически все: покупать товары или услуги, искать информацию о незнакомой местности, общаться с представителями компаний, и так далее. Не удивительно, что они стали неотъемлемой частью жизни каждого.

Такое распространение социальных сетей привлекло в них множество компаний, производящий какой-либо контент, ведь социальные сети стали очень удобным инструментом для его доставки. В том числе, многие новостные компании переориентировались на социальные сети, где они получают гораздо больше просмотров и откликов. С развитием социальных сетей появилась даже новая отрасль - *интернет-медиа* - компании, которые занимаются журналистикой, не выпуская печатные издания или телевизионные передачи, выпуская свои материалы исключительно в социальных сетях, или же на своих веб-сайтах.

На сегодняшний момент, большинство новостей как локального, так и глобального масштаба люди узнают прежде всего из новостных лент социальных сетей, которыми они пользуются. Часто это сообщения от официальных аккаунтов известных новостных агенств и изданий. Но в социальных сетях также публикуется информация от очевидцев событий, людей, которые оказались рядом с местом события или же являются непосредственными их участниками. С помощью данной информации новостные агенства и экстренные службы могут быстрее узнавать о каких-либо аномалиях или аварийных ситуациях, и оперативнее реагировать на происходящее.

В связи с этим, актуальной становится задача выделения аномальных и общественно значимых событий из потока сообщений в социальных сетях, а также их последующий анализ и выделение особенностей. Решение данной задачи сделает возможным, например, быстрое информирование окружающих о критической или аварийной ситуации.

В рамках данной выпускной квалификационной работы рассмотрены методы для сбора необходимой информации и её обработки, а также проведен сравнительный анализ методов выделения кластеров событий, соответствующих событиям в мире. Результаты данной работы предлагается использовать для построения системы детектирования событий в реальном времени.

Постановка задачи

Основная задача данной работы — оценка применимости существующих методов машинного обучения и кластеризации к решению задачи отделения уникальных кластеров сообщений, а также разработка программного комплекса для выделении общественно значимых событий на основе изученных методов.

Для достижения поставленной цели необходимо решить следующие задачи:

- Построение системы для сбора информации из потока сообщений в социальной сети Твиттер, а также системы хранения полученной информации.
- Предварительная обработка текстовой информации и приведение её к виду, с которым способны работать наиболее распространенные библиотеки машинного обучения и кластеризации.
- Изучение и поиск наиболее подходящего метода кластеризации полученных сообщений и выделение общественно значимых событий.
- Разработка метода ранжирования выделенных кластеров для получения наиболее значимых событий.
- Изучение возможности выделение событий в режиме реального времени.

Для решения поставленных задач была составлена схема решения, состоящая из следующих шагов:

- Разработка программного продукта для сбора сообщений и их доставки в систему хранения
- Развертывание и настройка системы хранения

- Запуск разработанных систем и сбор данных
- Обработка полученных данных
- Анализ различных методов кластеризации сообщений и выбор наилучшего для данной задачи
- Применение метода для получения кластеров, анализ и обработка полученных результатов

Описание предметной области

Обработка естественного языка

Естественный язык — язык, который используют люди для общения между собой. И хотя на компьютере можно очень быстро и эффективно производить численные и математические вычисления, автоматическая обработка естественного языка - одна из труднейших алгоритмических задач. Это связано с большим количеством факторов.

Основная проблема - отсутствие четких правил и ограничений при составлении фраз и предложений, которые отличаются от языка к языку. И хотя существует много правил построения предложений, написания слов или использования синтаксических конструкций в тех или иных ситуациях, в них нередко встречаются исключения или особые случаи написания. Это не позволяет использовать одни и те же правила разбора предложения на составляющие части в разных случаях, ведь необходимо делать поправку на контекст документа.

Другая проблема - неоднозначность отдельно взятого слова или словосочетания. Неопределенность возникает из-за того, что часто значение какой-то выделенной лексемы зависит от контекста предложения, в котором она расположена. Например, если в предложении встретилось слово *стекло,* то это может быть либо, существительное *стекло* в именительном падеже, либо глагол *стечь* в прошедшем времени. Без контекста предложения не всегда ясно, какой из этих двух вариантов выбрать. Также существуют неопределенности такого рода, для которых даже информация о контексте не дает возможность точно определить исходную форму слова.

На сегодняшний день, для обработки естественного языка используются методы машинного обучения, которые обучены на огромных наборах данных, размеченных вручную специалистами. Тем не менее, работа данных методов не является безошибочной, и требуется задание дополнительных правил для разбора особых случаев.

Выделение отдельных событий из множества сообщений

Возможность выделения сообщений об актуальных событий из потока сообщений позволит упростить и улучшить работу для новостных агентств и экстренных служб, позволяя им более оперативно получать информацию от очевидцев событий в реальном времени.

На сегодняшний день существует несколько методов выделения интересующих событий, которыми пользуются большинство пользователей и которые реализованы в интерфейсе большинства социальных сетей. В частности наиболее популярный способ — добавление *хэштегов* в сообщения. Хэштег — произвольный набор символов и цифр, начинающийся с символа «#». Данный метод позволяет пользователям объединять разные сообщения на одну тему с помощью добавление хэштега в текст сообщений.

Тем не менее, манипулирование хэштегами для проведения рекламных компаний и распространения спам сообщений — широко распространенная практика. Так как сообщения объединяются по хэштегам вне зависимости от основного текста сообщения, часто

можно встретить сообщения, не относящиеся к данной теме с тем же хэштегом. Также, данный метод требует от пользователей самостоятельно ставить хэштеги, а также знать о их существовании и правильный вариант их написания, хотя чаще всего пользователи не задумываются об этом и не ставят необходимые хэштеги.

Глава 1. Сбор сообщений.

Для исследования была выбрана социальная сеть Твиттер[1]. Это обосновывается следующим причинами:

- Поток сообщений содержит большое количество сообщений, особенно на русском языке
- Предоставляется удобный API для работы с потоком сообщений из сети
- Имеется ограничение на длину сообщений, из-за чего размер сообщения мало влияет на анализ

В то же время, нельзя не упомянуть о недостатках данной социальной сети, в частности большое количество «ботов» и «спамеров». Также, из-за ограничения на длину, некоторые сообщения не несут какой-либо смысловой нагрузки, и, с первого взгляда, больше похожи на набор слов.

1.1. Обзор API сети Твиттер

Программный доступ к сообщениям сети Твиттер производится с помощью специально разработанного API[2], к зависимости от типа которого можно получать доступ к разным данным. Всего существует три типа:

- Tweets API
- Streaming API
- Firehose API

Tweets API[3] позволяет получить доступ к сообщениям конкретных пользователей или группы пользователей. Взаимодействие с данным видом API осуществляется с помощью HTPP-запросов на

определенные *энд-поинты* — адреса, на которые необходимо отправлять GET или POST запросы для получения данных. В рамках данного API можно получать данные о конкретном пользователе, его общедоступной информации, а также его публичные сообщения или ответы на другие сообщения.

Этот вид API часто используется, для анализа сообщения и мнения конкретных пользователей, или всех пользователей относительно конкретного аккаунта или бренда.

Streaming API[4] позволяет получить доступ к потоку сообщений, отфильтрованных по некоторому признаку. В частности, с помощью данного API можно получить доступ к сообщениям, содержащим какой-то определенный термин, или написанных на заданном языке.

Взаимодействие с данным видом API реализовано на основе технологии WebSockets[5], которая позволяет установить постоянное двухстороннее клиент-серверное соединение, по которому производится обмен данными. Удобство данной технологии состоит в том, что после установки соединения сервер может сам отправлять данные клиенту, не дожидаясь запроса. В рамках данной системы, это позволяет серверу Streaming API самостоятельно отправлять сообщения клиентам, с которыми у него установлено соединение, в виде потока сообщений, что позволяет организовывать их обработку в режиме реального времени.

Firehose API [6] позволяет получить доступ к большинству сообщений, которые написаны в сети Твиттер. Данный вид API также реализован на основе технологии WebSockets. Он удобен, если необходимо собрать как можно больше данных. Но к сожалению, существует ряд ограничений:

- Данный вид API не является открытым, и для получения доступа к нему необходимо связываться с компанией Твиттер, и обосновывать получение доступа.
- Доступ к данному API является платным
- Невозможно настроить фильтрацию сообщений.

Для исследования был выбран Streaming API. Для получения доступа к данному API было необходимо создать аккаунт на сайте https://twitter.com, затем создать приложение на сайте https://apps.twitter.com/, используя созданный аккаунт. После регистрации приложения будут генерируются соответствующие ключи подписи и ключи доступа, которыми необходимо сопровождать запросы, направляемые в API.

Необходимые энд-поинты для HTTP и WebSockets запросов для каждого действия описаны в документации к API. Тем не менее, для популярных языков программирования написаны библиотеки-обертки данного API для упрощения работы. Данные библиотеки позволяют вместо HTTP запросов работать с встроенными объектами и структурами данного языка, при этом позволяя пользователю не задумываться о составлении и отправке запроса, а также обработке ответа. Для работы с этими библиотеками необходимо сообщить библиотеке свои данные для управления приложением, созданным в Твиттере, а затем вызвать необходимые методы, соответсвующие требуемым запросам.

1.2. Разработка программного продукта

В рамках данной работы программный продукт для сбора сообщений был разработан с использованием языка

программирования Python [7] версии 3.6. Выбор языка программирования был сделан на основе следующих факторов:

- Удобство языка для разработки «скриптовых» приложений, т.е. приложений без графического интерфейса, разработанных для определенной задачи сбора или обработки данных
- Наличие удобных библиотек для работы с Twitter API и системой хранения данных
- Наличие большого количества сервисов для удаленного размещения и работы программ в течении длительного времени

Реализованный программный продукт работал по следующему принципу:

- Установить соединение с Twitter Streaming API с использованием полученных ключей
- При получении нового объекта сообщения по внутренним параметрам проверялось, является ли это сообщение написанным на русском языке. Если нет, то сообщение отбрасывалось.
- Далее анализировалось сообщение на ненулевой отклик, то есть имеет ли сообщение хотя бы 1 «ретвит». Такое сообщение сохранялось в базе данных.

Исходя из этого алгоритма, программный продукт был разбит на три модуля:

1) Отвечающий за соединение с Twitter API
2) Фильтриерующий сообщения
3) Добавляющий сообщения в базу данных

Модуль, отвечающий за соединение разработан с использованием библиотеки tweepy [8], которая предназначена для работы с Twitter API. Она представляет собой удобную обертку над HTTP-запросами в API,

позволяя работать с объектами языка программирования, при этом генерируя и разбирая запросы и ответы от серверов Twitter API автоматически. Для работы с данной библиотекой необходимо подать ей в качестве параметров свои ключи доступа и верификации, и затем реализовать класс, наследуемый от одного и предоставленных классов, для получения доступа к API.

Так как в рамках данной работы для сбора сообщений был выбран Twitter Streaming API, был реализован класс, наследованный от tweepy.StreamListener, необходимый для подключения к данному виду API. В рамках реализации был разработан метод для обработки полученных сообщений, а также для обработки возникающих ошибок.

Модуль, отвечающий за фильтрацию сообщений был реализован в виде функции, принимающей объект сообщение и проверяющий необходимые признаки. Для определения языка, проверяется значение поля ‘lang’ в объекте сообщения, содержащего код языка в формате ISO-639-1[9], а также слова в сообщении. Для проверки отклика, проверяется специальное поля ‘retweets_count’ и ‘like_count’.

При прохождении фильтрации сообщение добавляется в систему хранилища с помощью соответсвующего модуля. Данный модуль извлекает следующую информацию о сообщении:

- Время написания
- Идентификатор сообщения (id)
- Число «ретвитов»
- Число отметок «нравится»
- Число ответов на данное сообщение в рамках диалогового функционала сети Твиттер

- Полный текст сообщения, включающий в себя ссылки на дополнительные материалы, такие как изображения, видео и сторонние ресурсы
- Время добавления сообщения в базу данных
- При обновлении числа «ретвитов» или отметок «нравится» - время последнего обновления

После извлечения информации создается внутренний объект сообщения, который конвертируется в формат JSON [10], и затем добавляется в базу данных.

1.3. Система хранения

В данной работе система хранения полученных сообщений была необходима для сохранения данных о полученных из потока сообщениях в течении длительного времени работы приложения для их сбора. В рамках данной работы, выбор был сделан в пользу использования нереляционной базы данных.

1.4. Нереляционные (NoSQL) базы данных

NoSQL[11] базы данных являются альтернативным подходом к организации хранения данных и доступа к ним. В отличии от реляционных баз данных, которые придерживаются требований ACID[12] - атомарность, консистентность, изолированность и надежность, нереляционные базы данных придерживаются другого набора требований BASE - базовая доступность, гибкое состояние и согласованность в конечном счете.

Данные свойства позволяют NoSQL базам данных быть лучше приспособленными к работе в высоконагруженных системах, таких как поисковые сервисы или сервисы хранения и управления пользовательским контентом. Также отсутствие ограничений реляционных баз данных позволяют легче масштабировать NoSQL системы хранения с помощью добавления новых ресурсов в систему, а также распараллеливания и балансировки запросов в разные экземпляры хранилища.

Тем не менее, сервисы, которые сильно зависят от атомарности и консистентности записи в базу данных, такие как банковские и биржевые сервисы, не могут использовать NoSQL системы, что является существенным ограничением.

1.5. Система хранения сообщений

При построении системы хранения на основе нереляционной базы данных была выбрана реализация NoSQL системы хранения MongoDB[13]. Она представляет собой документо-ориентированное хранилище, где единицей хранения является документ — иерархическая структура данных, а не строка в таблице. Такой выбор был сделан по следующим причинам:

- Для данной системы хранения не требуется задание схемы таблицы, то есть можно добавлять и рассматривать произвольные иерархические объекты в рамках одной коллекции документов
- Удобство добавления объектов сообщений в базу данных, так как сообщения, полученные из Twitter API представляют собой неупорядоченные и неструктурированные пары ключ-значения
- Простота работы с JSON-сериализованными объектами

- Высокая скорость работы даже при достаточно большом количестве записей

Особенностью хранилища MongoDB является возможность добавления BSON-объектков, которые представляют собой бинарное представление JSON-объектов. Данный факт позволяет MongoDB использовать меньше физического места на диске при хранении данных, а также более эффективно проводить поиск значений. При запросе в базу данных система автоматически преобразует полученный JSON в BSON.

Для добавления сообщений в базу данных в рамках работы приложения по сбору сообщений использовалась библиотека pymongo[14], представляющая собой набор функций и объектов для работы с коллекцией в базе данных MongoDB. Для работы данной библиотеке, необходимо указать хост и порт, на которых экземпляр базы данных принимает запросы, а затем, пользуясь методами библиотеки, делать запросы в базу данных.

Так как MongoDB не требует поддержания структуры документов, в параметры запроса можно положить почти любой JSON-объект. Пример запроса в базу данных с использование библиотеки pymongo, используемого в разработанной системе сбора и хранения сообщений выглядит так:

```
self.collection.find_one_and_update(
    {'id': retweetd[‘id_str']},
    {'$set': tweet},
    upsert=True
)
```

ПРИМЕР №1. ПРИМЕР ОБРАЩЕНИЯ В СИСТЕМУ ХРАНЕНИЯ С ПОМОЩЬЮ БИБЛИОТЕКИ PYMONGO

1.6. Организация работы системы сбора сообщений

Для сбора достаточного количества данных требовалось обеспечить бесперебойную работа программного продукта для сбора сообщении и базы данных для хранения полученной информации на протяжении длительного времени. На домашнем компьютере обеспечить такие условия практически невозможно. Поэтому, для обеспечения необходимого времени работы было решено использовать облачный сервис для размещения необходимых систем и программ.

Облачный сервис предоставляет собой набор сервисов для создания выделенных или виртуальных удаленных серверов, а также набор инструментов для их управлением и доступом к их данным. Современные провайдеры облачных услуг предоставляют возможности для создания и управления большим количеством облачных серверов, а также API для их управлением и конфигурацией. Также они предоставляют доступ к удаленным системам хранения данных и другим сервисам для работы с удаленными серверами.

В качестве провайдера облачного сервера был выбран сервис DigitalOcean[15]. Это одно из самых популярных на сегодняшний день решений для создания облачных серверов. Выбор в пользу данного провайдера был сделан по следующим причинам:

- Предоставление наиболее удобных методов взаимодействия с сервером
- Наличие большого количества готовых конфигураций для быстрого запуска и настройки удаленного сервера

- Наиболее полная и понятная документацией с описанием стандартных и нестандартных процессов настройки удаленных сервисов

Для организации работы приложения был выбран вариант облачного сервера с 4 GB оперативной памяти, 2 виртуальных CPU, 80 GB SSD. На сервер с помощью системы контроля версий был загружен разработанный программный продукт, а также установлена и настроена система хранения полученных сообщений. Также, были установлены все необходимые программы, библиотеки и расширения для корректной работы разработанной системы.

Система сбора сообщений работала в течении двух периодов:

- с 2 по 13 апреля 2018 г.
- с 27 апреля по 7 мая 2018г.

За время работы системы было собрано около 424,000 сообщений на русском языке, на которых в последствии и проводились анализ и сравнение алгоритмов кластеризации.

Глава 2. Обработка собранных данных

Перед проведением исследования и сравнения различных алгоритмов кластеризации необходимо привести собранные сообщения к векторному представлению, чтобы уже по их векторному виду проводить кластеризацию и анализ проведенной кластеризации.

2.1. Приведение слов к нормальной форме

При обработке естественного языка всегда возникает проблема наличия разных форм слова в одной и той же коллекции документов, которые мы должны учитывать как одно слово, ведь разность формы слова почти не меняет смысл слова или его значение, а меняет только временной аспект или носителя действия или признака. Следовательно, при обработке естественного языка необходимо применять методы для того, чтобы много разных форм слова стали одним. Для этого чаще всего каждое слово приводят в нормальную или словарную форму слова. Для большинства существительных это единственное число и именительный падеж, хотя встречаются и слова, для которых нормальная форма отличается, например слова, которые не употребляются в единственном числе.

В данной работе приведение слов в нормальную форму производилось с помощью библиотеки pymorphy2[16][17] для языка Python 3. Данная библиотека реализована на основе словаря, где слова объединены группы по слову в нормальной форме, при этом каждому слову в виде тэгов назначены дескрипторы данной формы слова и общие дескрипторы лексемы. В частности, для слова *ёж* данная группа слов выглядит следующим образом:

СЛОВО	ЧАСТЬ РЕЧИ	ОДУШЕВЛЕННОЕ/ НЕОДУШЕВЛЕННОЕ	РОД	ЧИСЛО	ПАДЕЖ
ЁЖ	NOUN	ANIM	MASC	SING	NOMN
ЕЖА	NOUN	ANIM	MASC	SING	GENT
ЕЖУ	NOUN	ANIM	MASC	SING	ACCS
ЕЖА	NOUN	ANIM	MASC	SING	ACCS
ЕЖОМ	NOUN	ANIM	MASC	SING	ABLT
ЕЖЕ	NOUN	ANIM	MASC	SING	LOCT
ЕЖИ	NOUN	ANIM	MASC	PLUR	NOMN
ЕЖЕЙ	NOUN	ANIM	MASC	PLUR	GENT
ЕЖАМ	NOUN	ANIM	MASC	PLUR	DATV
ЕЖЕЙ	NOUN	ANIM	MASC	PLUR	ACCS
ЕЖАМИ	NOUN	ANIM	MASC	PLUR	ABLT
ЕЖАХ	NOUN	ANIM	MASC	PLUR	LOCT

ТАБЛИЦА №1. ПРИМЕР ГРУППЫ СЛОВ ДЛЯ РАЗБОРА НОРМАЛЬНОЙ ФОРМЫ В БИБЛИОТЕКЕ PYMORPHY2

В каждой таблице первым всегда находится нормальная форма для данного слова. По таким таблицам возможен эффективный поиск нормальной формы для слова. Для этого необходимо найти целевое слово в словаре, затем по его группе найти нормальную форму для данного слова. Для эффективного поиска слов в словаре в реализации данной библиотеки используется детерминированный конечный автомат, составленный из всех слов в словаре вместе с закодированной информацией о группе данного слова.

2.2. Bag-of-Words и Word2Vec модели

Приведение естественного языка, слов и предложений к виду, с которыми могут работать существующие алгоритмы машинного

обучения, то есть переход от слов и документов к их векторному представлению - задача которую необходимо решать всем исследователем, занимающимся анализом текстовой информации. На сегодняшний день существует несколько подходов для приведения текстов в векторную форму.

Наиболее простой для реализации и использования — Bag-of-Words[18] представление. Это один из самых распространенных алгоритмов приведения документов к векторному виду. Данный алгоритм устроен следующим образом:

1. Разбить все документы, представленные в коллекции, на отдельные слова или другие интересующие токены
2. Составить словарь из всех уникальных токенов
3. Каждому документу сопоставить вектор - $(x_1, x_2, \ldots, x_n)$, где $x_i = 1$, если i-ое слово из словаря присутствует в данном документе, 0 - иначе.

Такое представление имеет преимущества и недостатки. В частности, реализация данного преобразования — очень простая задача, и использование данного предоставления в задачах классификации показывает неплохие результаты. Тем не менее, имеются так же и существенные проблемы:

- Отдельные токены и слова не зависят друг от друга, что не совсем корректно для слов в естественном языке
- Нельзя установить функцию для «похожести» слов, их близости между друг другом из-за того, что слово представлено индексом в словаре, хотя в естественном языке есть такие вещи, как синонимы и антонимы

Данные ограничения bag-of-words представления делают успешную кластеризацию сообщений очень непростой задачей, так как для кластеризации необходимо устанавливать значение «похожести» между словами и документами. По этим причинам было принято решение отказаться от использовании bag-of-words модели при дальнейшем исследовании.

Другим популярным на сегодняшний день набором алгоритмом приведения документов естественного языка к векторному виду является Word2Vec[19] алгоритм. В рамках данного алгоритма были предложены две архитектуры нейронных сетей для построения векторного представления слов с учетом контекста и окружающих слов.

Первая архитектура, которая называется Skip-gram, основаная на обучение нейронной сети, где на вход подается целевое слово, для которого хотим получить векторное представление, а выходом являются слова, окружающие целевое слово. При этом, на входе и на выходе слова представлены в виде векторов составленных с помощью bag-of-words модели, то есть для каждого слова сопоставлен вектор размерности $1 \times N$, где N — общее число уникальных слов в коллекции документов, где везде стоят 0, кроме позиции i, где i - номер рассматриваемого слова в словаре. Таким образом, на вход подается вектор размерности $1 \times N$, на выходе получается вектор - $1 \times (R \cdot N)$, где R — размер окна вокруг слова, то есть количество рассматриваемых слов, окружающих целевое слово.

Нейронная сеть также содержит один внутренний слой, размерность которого равна $1 \times V$, где V - желаемый размер

векторного представления слова. В данном случае параметр V является параметром модели, который можно изменять для лучшего представления слова. Коэффициенты внутреннего слоя после обучения модели являются искомым векторным представлением нашего целевого слова.

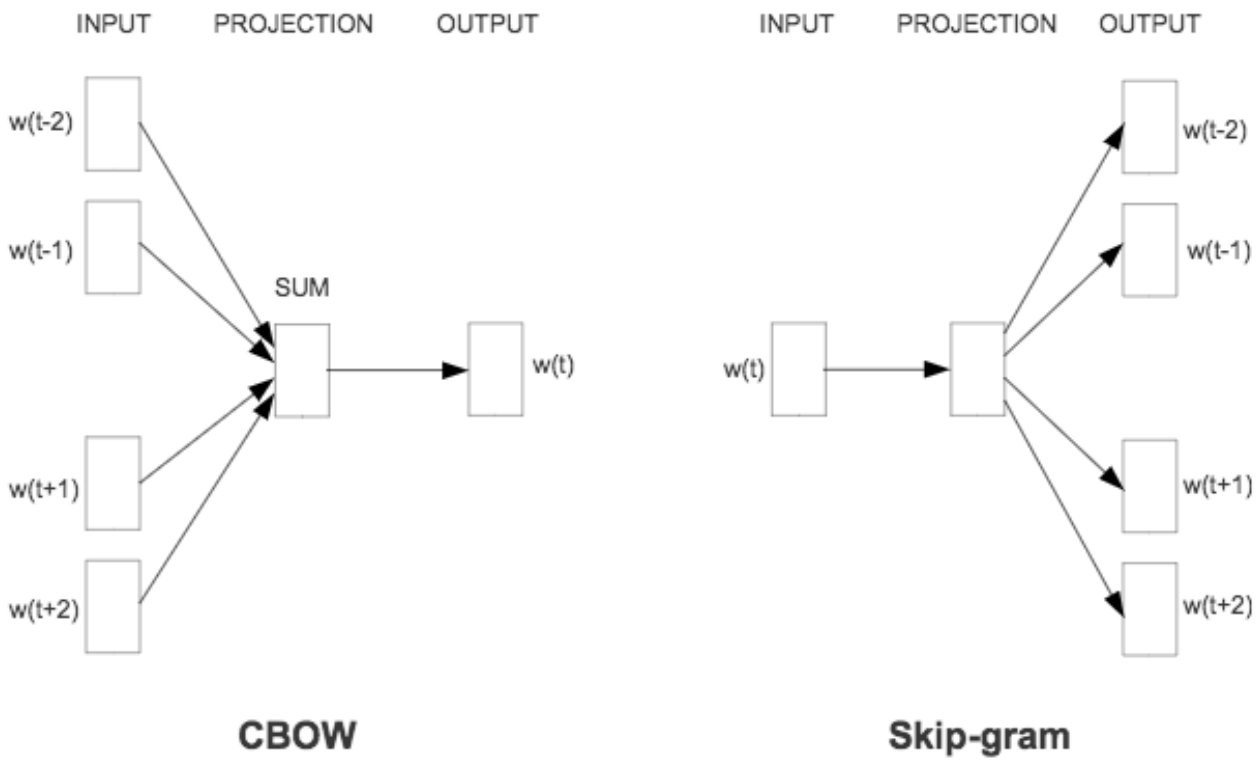

РИСУНОК №1. ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ АРХИТЕКТУР НЕЙРОННОЙ СЕТИ, ИСПОЛЬЗУЕМЫМИ В АЛГОРИТМЕ WORD2VEC.

Вторая архитектура носит названия Continious Bag-of-Words (CBOW). Архитектуры построения сетей в CBOW и Skip-gram достаточно схожи между собой, с той разницей, что в CBOW поменяны местами вход и выход. На вход CBOW мы подаем вектор $1 \times (R \cdot N)$, где R — размер окна вокруг целевого слова. На выходе получаем $1 \times N$ вектор выходного слова. В этой архитектуре также

присутствует один скрытый слой размерности $1 \times V$, где V - желаемый размер векторного представления слова, коэффициенты которого и будут искомым представлением слова.

Основная разница между двумя подходами это способ генерации векторного представления слова. В CBOW модели мы предсказываем слово по контексту, который наблюдаем в данный момент, в то время как в Skip-gram модели мы предсказываем контекст по слову. Но несмотря на различия, на практике эти две модели показывают почти идентичные результаты на одинаковых данных. Исключением являются только слова, редко встречающиеся в рассматриваемой коллекции документов. Из-за описанных разностей подходов двух алгоритмов, Skip-gram лучше представляет редкие слова, по сравнению с Continious-Bag-of-Words. Графически обе архитектуры представлены на рисунке №1.

2.3. Именованные сущности

Именованные сущности — объекты реального мира, которые могут быть обозначены именами собственными или который может быть отнесен к заранее обозначенным категориям, таким как: «Влиятельные персоны», «Организации» и т.п. В естественном языке достаточно часто встречаются разные именованные сущности, и во многих случаях они достаточно сильно влияют на смысл сказанного. Тем не менее, выделение именованных сущностей — сложная задача, которая не может быть полностью решена на сегодняшний день. Основаная проблема решения данной задачи заключается в том, что для выделения именованных сущностей из текста необходимо обладать большим количеством знаний не только о структуре предложений и текста, но и о ситуации в мире, существующих компаниях и людях.

В данный момент для выделения именованных сущностей используют два основных подхода. Первый - составление правил разбора основанных на грамматических особенностях языка и обработки выделенных сущностей для выделения необходимой информации. Этот подход характеризуется хорошей точностью, тем не менее методы на его основе плохо переносятся на выделение сущностей из текстов другой тематики, и часто разработка таких методов занимает огромное количество времени и требует большого количества специалистов. Другой подход основан на методах машинного обучения и построении статистических моделей. Данный подход не требует больших временных затрат, но чувствителен к недостатку данных. Для получения хороших результатов для данного метода необходимо большое количество размеченных данных.

2.4. Обработка данных

В рамках обработки собранных сообщений были сделаны следующие шаги:

1. Из системы хранения были импортированы данные о собранных сообщениях в формате JSON
2. Из импортированных данных были извлечены тексты сообщений, из которых была создана коллекция документов для последующего анализа
3. Все слова в сообщениях были приведены в нормальную форму, исключены предлоги, междометия и другие *стоп-слова* - слова не несущие важной смысловой нагрузки
4. На основе полученной коллекции документов были созданы два представления данных на основе Word2Vec: основанный на всех

словах, которые встречаются в сообщениях, и на основе только именованных сущностей, встречающихся в сообщениях

Обработка сообщений проводилась в среде разработки Jupyter Notebook[20] с использованием языка программирования Python 3.6. Преимуществом данной среды разработки является возможность подключения к удаленным серверам, на которых расположены файлы с исходным кодом и другие ресурсы, а также возможность удаленной работы с данными файлами и ресурсами.

В рамках данной работы на облачном сервере, предназначенном для сбора данных была установлена среда Jupyter Notebook, и далее вся работа с данными проводилась путем подключения с рабочего компьютера к удаленному серверу. Благодаря этому, отпала необходимость в переносе и копировании собранных данных, а также появилась возможность продолжать сбор данных, последовательно добавляя новые сообщения к уже собранным.

В рамках обработки данных было решено рассмотреть два варианта преобразования сообщений к векторному виду. Первый вариант - проводить векторное преобразование для всех слов в сообщениях. Построение такой модели представления позволяет так или иначе учитывать все слова в сообщении при построении его векторного представления. Тем не менее, в рамках рассматриваемой задачи выделение актуальных событий была выдвинута гипотеза о том, что информация о всех словах может быть избыточной и плохо влиять на результаты кластеризации, и соотвественно, выделения событий.

Для проверки данной гипотезы была построена вторая модель. В рамках данной модели из всех сообщений отбирались лишь

существительные и именованные сущности, и векторное представление строилось лишь для этих слов. Затем, векторное представление сообщений строилось на основе векторного представления существительных и именованных сущностей, встречающихся в данном сообщении. Выделение части речи для каждого слова проводилась при помощи библиотеки pymorphy2, а извлечение именованных сущностей — с помощью библиотеки Natasha[21].

Векторное представление для сообщений строилось путем усредненных значений векторов для слов, рассматриваемых в данной модели. Для первой модели учитывались все слова в сообщении, для второй — только существительные и именованные сущности.

Глава 3. Анализ методов кластеризации

В рамках сравнительного анализа были рассмотрены четыре алгоритма кластеризации, которые можно разделить на две группы:

- Не требующие задания числа кластеров
  - DBSCAN
  - Affinity Propagation
- Требующие задания кластеров
  - k-means
  - Иерархическая кластеризация

3.1. DBSCAN[22]

Основная идея данного алгоритма — группировка точек пространства, которые расположены близко к друг другу, при этом отбрасывание отдельно лежащих точек как выбросов. При этом отличаются три вида точек: основные точки, достижимые точки и выбросы.

Будем называть точку *основной*, если внутри ϵ-шара с центром в данной точке лежит как минимум P точек, где ϵ, P - параметры алгоритма, которые мы задаем при инициализации процедуры. При этом, сама точка включается в число точек внутри шара.

Будем называть точку q *достижимой* из точки p, если она не является основной и выполняются следующие условия:

- Точка находится внутри ϵ-шара некоторой основной точки. Тогда такая точка будет называться достижимой напрямую

- Или существует путь $p_1, p_2, p_3, \ldots, p_n, p_1 = p, p_n = q$, где все точки на пути кроме может быть q - основные точки, и расстояние между p_{i+1} и $p_i \leq \epsilon$.

Все точки, не подпадающие под определения основной и достижимой точки являются выбросами.

Для формирования кластеров необходимо оценивать близость пары точек. Тем не менее, если они обе не являются основными у нас пока нет инструмента чтобы сказать, являются ли близкими между собой, то есть у нас есть основания добавить их в наш кластер, или же нет. Для этого, введем понятие близости или связанности двух точек. Будем называть две точки p, q связанными, если существует такая точка r, что p и q являются достижимыми точками из r.

Тогда будем называть кластером множество точек, для которых выполняются следующие свойства:

- Все точки состявляющие кластер являются связанными
- Если точка является достижимой из всех основных точек кластера, то она также находится в этом кластере

Исходя из этого, алгоритм для определения кластеров в DBSCAN можно записать следующим образом:

1. Задаем значения переменных ϵ, P
2. Находим всех ϵ соседей для каждой точки, и находим основные точки для нашего множества
3. Выделяем кластеры, находя точки, связанные с основными точками, и затем составляя кластеры по описанному правилу

4. Точки, которые не были отнесены ни к одному кластеру будем считать выбросами и не относить ни к какому

3.2. Affinity Propagation[23]

Данный метод основан на концепции передачи сообщений между объектами. В данном алгоритме мы рассматриваем множество точек $x_1, x_2, \ldots, x_n$, для которых задана функция $s(x_i, x_j)$, которая представляет собой уровень похожести между двумя элементами нашего множества. Чаще всего выбирают симметричные функции похожести, то есть такие, что $s(x_i, x_j) = s(x_j, x_i)$.

При этом также можно рассматривать значение $s(x_i, x_i)$. Для алгоритма Affinity Propagation это значение будет означать, насколько i -ый объект может быть *образцом,* то есть наиболее характерным элементом, для своего кластера. Эти значения являются параметром алгоритма, и влияют на его работу следующим образом:

- Чем меньше данные значения, тем менее вероятно, что рассматриваемый объект являет образцом кластера, тем меньше кластеров будет выделено из данных
- Чем выше значения, тем больше кластеров будет выделено

В работе алгоритма также используется две матрицы:

- Матрица отклика - R, значение которой в строке i и столбце j: $r(i, j)$ показывает, насколько хорошо объект j подходит для роли образца для объекта i, то есть насколько хорошо объект j характеризует образец i.

- Матрица доступности - A, в строке i и столбце j: $a(i,j)$ показывает, насколько удачным будет выбор объекта j в качестве образца для объекта i.

Данные матрицы инициализируются нулями.

Далее алгоритм итеративно обновляет значения матриц следующим образом:

1. Обновление значений для матрицы отклика

$$r(i,k) = s(i,k) - \max_{k' \neq k} \left[a(i,k') + s(i,k') \right]$$

2. Обновление значений для матрицы доступности

$$a(k,k) = \sum_{i' \neq k} \max(0, r(i',k))$$

$$a(i,k) = \min(0, r(k,k) + \sum_{i' \notin \{i,k\}} \max(0, r(i',k)), for\ i \neq k$$

Данные обновления проводятся, пока не будет проведено максимальное количество итераций, что является параметром, который задается при инициализации модели, или если процедура не изменяет значения больше, чем заданное маленькое число ϵ.

3.3 Kmeans[24]

Данный метод является наиболее популярным методом кластеризации на сегодняшний день. Его алгоритм основан на

движении центров кластеров ближе к истинному центру между точками, который он объединяет. При этом всего рассматривается k точек - центров кластеров. Параметр k является настраиваемым параметром.

Формально, мы ставим следующую задачу оптимизации:

$$(\mu_1, \ldots, \mu_k) = \arg\min_S \sum_{i=1}^{k} \sum_{x \in S_i} dist(x, \mu_i)$$

где $dist(x, \mu_i)$ - значение расстояния между точками x, μ_i. В зависимости от задачи можно выбирать разные виды функции $dist$.

Тогда, алгоритм для работы Kmeans будет состоять из поочередного выполнения следующих шагов:

1. Присваивание: на основе текущего положения центроидов вычисляются кластеры:

$$S_i = \{x : dist(x, \mu_i) \leq dist(x, \mu_j) \forall j : 1 \leq j \leq k\}$$

2. Обновление: на основе собранных кластеров пересчитывается положение центроидов:

$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$$

Данная итеративная процедура продолжается, пока число итераций не достигло заданного числа максимального количества итераций, что является параметров модели, или же пока положение центроидов изменяется достаточно сильно, то есть изменение какого-то центроида больше порогового значения.

3.4. Иерархическая кластеризация

Данный метод основан на последовательном объединении кластеров, которые наиболее близко расположены друг к другу. Изначально, каждому объекту приписывается свой кластер. Далее, происходит процедура поиска наиболее близко расположенной пары кластеров, и производится процедура их объединения.

Объединять кластеры можно на основе попарных расстояний между наблюдениями из двух кластеров, или же на основе критериев связи между кластерами, которые характеризуют близость не наблюдений, а объединены кластеров. Типичные метрики и критерии связи для алгоритма иерархической кластеризации показаны далее.

Типичные метрики для расчета расстояния между точками:

Евклидово расстояние $||a-b||_2 = \sqrt{\sum_i (a_i - b_i)^2}$

Манхеттен расстояние $||a-b||_1 = \sum_i |a_i - b_i|$

Максимальное расстояние: $||a-b||_\infty = \max_i |a_i - b_i|$

Типичные формулы для критериев связи между двумя кластерами:

Метод полной связи: $dist(A,B) = \max\{d(a,b) : a \in A, b \in B\}$

Метод неполной связи: $dist(A,B) = \max\{d(a,b) : a \in A, b \in B\}$

3.5. Silhouette coefficient

Для сравнения качества кластеризации разных алгоритмов используется метрика качества под названием silhouette coefficient[25]. Данная метрика была выбрана потому что для ее расчета не обязательно наличие верных значений кластеризации для объектов. Эта метрика характеризует консистентность и состоятельных полученных кластеров.

Для каждого объекта из набора данных коэффициент подсчитывается следующим образом:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}},$$

где $a(i)$ - среднее расстояние между объектом и всеми другими объектами в том же кластере

$b(i)$ - наименьшее среднее расстояние между объектом и всеми другими объектами в другом кластере

- Значение коэффициента ~ -1 означает, что данный объект должен принадлежать другому кластеру
- Значение коэффициента ~ +1 означает, что данный объект находится в правильном кластере
- Значение коэффициента ~ 0 показывает, что данный объект лежит на границе между двумя кластерами

Для получение значения silhouette coefficient для оценки проведенной кластеризации вычисляется среднее значение для всех объектов.

3.6. Интерпретация результатов

Для двух видов векторного представления собранных сообщений, полученных в результате преобразования данных, была проведена

кластеризация всеми четырьмя способами, то есть всего было получено 8 вариантов кластеризации. Для алгоритмов, где требуется подбор числа кластеров, подбор оптимального количества кластеров производился кластеризацией поиском числа кластеров, разбиение на которое давало наибольшее значение silhouette coefficient. Для проведения кластеризации были использованы библиотеки scikit-learn и pandas для языка Python. Scikit-learn[26] предоставляет реализованные алгоритмы машинного обучения для использования, а также большое количество функций и утилит для обработки данных и подсчета необходимых метрик, в том числе и silhouette coefficient. Pandas[27] позволяет удобно хранить и обрабатывать данные в форматах JSON, CSV и подготовить их для работы алгоритмов машинного обучения из scikit-learn.

Оценка и сравнение результатов разных алгоритмов проводилась прежде всего сравнением полученных silhouette coefficient для разбиений, а также просмотром наиболее релевантных кластеров и субъективной оценкой похожести сообщений, содержащихся в данном кластере. Сравнительные характеристики для разных алгоритмов кластеризации представлены в таблицах №2 и №3.

Для модели представления сообщений, содержащей все слова, изначально случайным образом было выбрано подмножество сообщений размеров 50,000 объектов сообщений, для которых была проведена процедура приведения к векторному виду, а затем проведена кластеризация полученных векторов. Данный шаг позволил сразу отбросить метод DBSCAN, так как он не смог выделить более одного кластера сообщений. Поэтому в дальнейшем, исследования данного метода не проводилось. Для других методов кластеризация

была проведена успешно, выделены кластеры и подсчитаны значения silhouette score.

Для модели, содержащей информацию только о именованных сущностях и существительных была проведена процедура с теми же шагами: сначала работа алгоритмов рассматривалась на подмножестве из 50,000 сообщений, затем на всем множестве собранных данных. При использовании данной модели представления метод DBSCAN также показал неудовлетворительные данные, выделив всего один кластер, и также был исключен из дальнейшего исследования. Для других методов значение silhouette coefficient было выше, чем при использовании данных о всех словах. Из этого можно сделать вывод, что для отделения кластеров сообщения информация о дейстиях может являться избыточной и добавляет шум в наблюдения.

	DBSCAN	AFFINITY PROPAGATION	KMEANS	ИЕРАРХИЧЕСКАЯ КЛАСТЕРИЗАЦИЯ
ВРЕМЯ РАБОТЫ АЛГОРИТМА	15 МИН	30 МИН	4 МИН/ОДНА КЛАСТЕРИЗАЦИЮ	6 МИН/ОДНА КЛАСТЕРИЗАЦИЯ
СРЕДНЕЕ КОЛИЧЕСТВО ИСПОЛЬЗУЕМОЙ ОПЕРАТИВНОЙ ПАМЯТИ	5,7 Гб	3,9 Гб	5,7 Гб	6,8 Гб
ЧИСЛО ВЫДЕЛЕННЫХ КЛАСТЕРОВ НА ПОДМНОЖЕСТВЕ СООБЩЕНИЙ РАЗМЕРА 50000	1 кластер	319	420	388
ЧИСЛО ВЫДЕЛЕННЫХ КЛАСТЕРОВ НА МНОЖЕСТВЕ ВСЕХ СООБЩЕНИЙ		2102	1528	1972
SILHOUETTE SCORE ДЛЯ ПОДМНОЖЕСТВА СООБЩЕНИЙ РАЗМЕРА 50000	0	0,1578	0,057	0,0498
SILHOUETTE SCORE ДЛЯ КЛАСТЕРИЗАЦИИ ВСЕХ СООБЩЕНИЙ		0,1548	0,058	0,0501

ТАБЛИЦА №2. СРАВНИТЕЛЬНЫЕ ПОКАЗАТЕЛИ АЛГОРИТМОВ ПРИ КЛАСТЕРИЗАЦИИ НА ОСНОВЕ WORD2VEC МОДЕЛИ, СОДЕРЖАЩИЕ ДАННЫЕ О ВСЕХ СЛОВАХ

По результатам сравнения рассмотренных методов кластеризации можно сделать вывод, что из рассмотренных методов наиболее подходящим для задачи можно назвать метод Affinity Propagation. При его использовании silhouette coefficient получился наибольшем, что свидетельствует о большей консистентности кластеров, а также он не требует больших временных затрат на поиски наиболее релевантных кластеров, как методы Kmeans и иерархической кластеризации.

	DBSCAN	AFFINITY PROPAGATION	KMEANS	ИЕРАРХИЧЕСКАЯ КЛАСТЕРИЗАЦИЯ
ВРЕМЯ РАБОТЫ АЛГОРИТМА	10 МИН	25 МИН	3 МИН/ОДНА КЛАСТЕРИЗАЦИЮ	4 МИН/ОДНА КЛАСТЕРИЗАЦИЯ
СРЕДНЕЕ КОЛИЧЕСТВО ИСПОЛЬЗУЕМОЙ ОПЕРАТИВНОЙ ПАМЯТИ	5,1 Гб	3,4 Гб	5,0 Гб	6,3 Гб
ЧИСЛО ВЫДЕЛЕННЫХ КЛАСТЕРОВ НА ПОДМНОЖЕСТВЕ СООБЩЕНИЙ РАЗМЕРА 50000	1 кластер	357	455	429
ЧИСЛО ВЫДЕЛЕННЫХ КЛАСТЕРОВ НА МНОЖЕСТВЕ ВСЕХ СООБЩЕНИЙ		2314	1899	2140
SILHOUETTE SCORE ДЛЯ ПОДМНОЖЕСТВА СООБЩЕНИЙ РАЗМЕРА 50000	0	0,1724	0,061	0,0607
SILHOUETTE SCORE ДЛЯ КЛАСТЕРИЗАЦИИ ВСЕХ СООБЩЕНИЙ		0,1725	0,061	0,0609

ТАБЛИЦА №3. СРАВНИТЕЛЬНЫЕ ПОКАЗАТЕЛИ АЛГОРИТМОВ ПРИ КЛАСТЕРИЗАЦИИ НА ОСНОВЕ WORD2VEC МОДЕЛИ, СОДЕРЖАЩИЕ ДАННЫЕ ОБ ИМЕНОВАННЫХ СУЩНОСТЯХ И СУЩЕСТВИТЕЛЬНЫХ

Субъективное рассмотрение некоторого количества кластеров, полученных в результате работы метода Affinity Propagation показывает, что кластеры, сообщения в которых более «похожи» друг на друга, часто содержат хорошие структурированные сообщения, опубликованные от аккаунтов новостных изданий и СМИ, при этом

данные кластеры включают сообщения обычных пользователей на тематику данного события. Тем не менее, присутствует много кластеров, содержащих сообщения, которые нельзя отнести ни к какой теме, или же сообщения, содержащие спам или рекламу.

Выводы

В данной работе в рамках задачи выделения актуальных событий из потока сообщений был создан программный продукт для сбора сообщений из сети «Твиттер», организована система хранения полученных сообщений а также организована работа программного продукта в течении длительного времени. Также были опробованы два вида векторного представления полученных текстовых сообщений и проведен сравнительный анализ известных методов кластеризации для выделения кластеров сообщений, объединенных одной темой.

По результатам проведенного анализа был сделан вывод, что для данной задачи наиболее удачным представлением является векторное представление, учитывающее только именованные сущности в тексте сообщения. Наиболее эффективном методом кластеризации из рассмотренных был принят Affinity Propagation на основании наибольшего значения silhouette coefficient.

Полученные кластеры сообщений, в которых содержались хорошо структурированные сообщения от официальных аккаунтов СМИ, содержали достаточно похожие сообщения, в том числе от обычных пользователей.

Тем не менее, присутствовали кластеры, содержащие разрозненные сообщения. В данных кластерах содержались сообщения исключительно от обычных пользователей сети, и часто не выражали никакой мысли или содержали спам или рекламу. Также встречались сообщения, которые просто содержали набор слов или букв. Для более эффективной работы системы кластеризации необходимо отсеивать и фильтровать данные сообщения.

Тем не менее, проведенный анализ позволил выбрать наиболее эффективный метод для выделения актуальных тем. На основе этой работы можно реализовывать автоматическую систему выделения актуальных сообщений, собирающих информацию в реальном времени.

Список литературы

1. Твиттер. https://twitter.com/
2. Twitter Developer Platform — Twitter Developers. https://developer.twitter.com/
3. Search Tweets — Overview. https://developer.twitter.com/en/docs/tweets/search/overview
4. Filter Realtime Tweets — Overview. https://developer.twitter.com/en/docs/tweets/filter-realtime/overview
5. RFC 6455 — The WebSocket Protocol. https://tools.ietf.org/html/rfc6455
6. Firehose API — Compliance. https://developer.twitter.com/en/docs/tweets/compliance/api-reference/compliance-firehose.html
7. Python. https://www.python.org/
8. Tweepy. https://github.com/tweepy/tweepy
9. ISO-639-1. https://www.iso.org/standard/22109.html
10. RFC 8259 — The JavaScript Object Notation (JSON) Data Interchange Format. https://tools.ietf.org/html/rfc8259
11. Strauch, Christoph. "NoSQL whitepaper" [PDF]. Stuttgart: Hochschule der Medien. www.christof-strauch.de/nosqldbs.pdf
12. Haerder, T.; Reuter, A. (1983). "Principles of transaction-oriented database recovery". ACM Computing Surveys. 15 (4): 287. doi: 10.1145/289.291
13. MongoDB. https://www.mongodb.com/
14. PyMongo — the Python driver for MongoDB. https://github.com/mongodb/mongo-python-driver
15. DigitalOcean: Cloud Computing. https://www.digitalocean.com/
16. Морфологический анализатор pymorphy2. https://pymorphy2.readthedocs.io/en/latest/index.html
17. Korobov M.: Morphological Analyzer and Generator for Russian and Ukrainian Languages // Analysis of Images, Social Networks and Texts, pp 320-332 (2015).
18. G. Salton , A. Wong , C. S. Yang, A vector space model for automatic indexing, Communications of the ACM, v.18 n.11, p.613-620, Nov. 1975
19. Mikolov T. et al. Efficient estimation of word representations in vector space //arXiv preprint arXiv:1301.3781. – 2013.
20. Project Jupyter. http://jupyter.org/

21. Natasha: Rule-based named entity recognition library for russian language. https://github.com/natasha/natasha
22. Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M., eds. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231. CiteSeerX 10.1.1.121.9220 . ISBN 1-57735-004-9.
23. Brendan J. Frey; Delbert Dueck (2007). "Clustering by passing messages between data points". Science. 315 (5814): 972–976. doi: 10.1126/science.1136800. PMID 17218491
24. Lloyd., S. P. (1982). "Least squares quantization in PCM". IEEE Transactions on Information Theory. 28 (2): 129–137. doi:10.1109/TIT.1982.1056489.
25. Peter J. Rousseeuw (1987). "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis". Computational and Applied Mathematics. 20: 53–65. doi:10.1016/0377-0427(87)90125-7
26. Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
27. Python Data Analysis Library. https://pandas.pydata.org/