МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программной и системной инженерии

РЕКОМЕНДОВАНО К ЗАЩИТЕ
В ГЭК И ПРОВЕРЕНО НА ОБЪЕМ
ЗАИМСТВОВАНИЯ
Заведующий кафедрой
д.т.н., профессор
А.Г. Ивашко
20.06 2019 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(магистерская диссертация)

ОЦЕНКА ДОСТОВЕРНОСТИ СЛУХОВ В СРЕДСТВАХ МАССОВОЙ ИНФОРМАЦИИ

Прикладная информатика 09.04.03
Магистерская программа: Прикладная информатика в экономике

Выполнил работу
Студент 2 курса
очной формы обучения
Черняев
Александр
Андреевич

Научный руководитель
д.т.н., профессор
Ивашко
Александр
Григорьевич

Рецензент
к.фил.н., доцент
Бидуля
Юлия
Владимировна

г. Тюмень, 2019

РЕФЕРАТ

Аннотация. Рассматривается метод проверки на наличие слуха в средствах массовой информации. Для решения данной задачи исследованы различные модели оценки поведения пользователя, анализ текста сообщения, а также определение временных показателей распространения сообщений. Определены основные параметры, которые необходимы для определения является ли текст слухом. За время выполнения задачи были проанализированы популярные инструменты для получения данных из социальных сетей, а также вручную составлена и промаркирована выборка для обучения нейронной сети. Проведены множественные проверки моделей нейронной сети для получения наиболее корректного результата. Анализ модели выполнен с применением таких метрик как accuracy, recall, precision и F1 оценок. Так же в ходе выполнения исследуется задача о доверии к пользователю.

Глава 1. Аналитический обзор. В данной главе приводится анализ существующих информационных систем, работ посвящённых данной теме и другие материалы, которые могут быть использованы для решения поставленных целей. Описываются причины, по которым были выбраны методы, по которым определяется наличие слуха в сообщении.

Глава 2. Методологии и средства разработки. Данная глава рассматривает методы, по которым будет осуществляться проверка на наличие слуха.

Глава 3. Исследование предметной области и определение функциональных требований к системе. Описывает методы, при помощи которых будет создано приложение. Описание библиотек, базы данных и другие средства разработки программного обеспечения.

Глава 4. Архитектура системы и описание ее основных компонентов. Описывает создание программного обеспечения в виде консольного приложения, библиотеки и создание графического интерфейса.

Сведения об объеме отчета: 83 страницы.

Количество иллюстраций: 18 шт.

Количество таблиц: 6 шт.

Количество частей отчета: 4 главы, введение, вывод и список литературы.

Количество источников: 34 источника.

Ключевые слова: искусственный интеллект, слух, нейронная сеть, сбор данных, анализ текста, оценка поведения пользователя.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

За последние 15 лет интернет стал одним из основным источником новостей для большинства людей на всей планете. Совсем недавно появившиеся и набравшие огромную популярность социальные сети и сетевые сервисы, такие как Twitter, ВКонтакте, Одноклассники и т.д., сильно повлияли на новостные агрегаты и на всю область журналистики. Теперь больше, чем когда-либо, люди обращаются к социальным медиа как к источнику новостей это особенно наблюдательно для ситуаций с новостями, когда люди жаждут быстрых обновлений по какой-либо цепочке событий. Более того, вездесущность, доступность, скорость и простота использования социальных сетей сделали их бесценными источниками информации из первых рук.

Этот беспрецедентный переход от традиционных средств массовой информации, где есть четкое различие между журналистами и новостными потребителями, в социальные сети, где новости публикуются в толпе и любой может быть репортером. Это добавляет множество проблем для различных слоев общества, таких как журналисты, аварийные службы и новостные потребители. Журналисты в современном мире должны конкурировать с огромным потоком данных обычных пользователей, из-за этого основным фактором качества становится время, за которое была опубликована новостная статья. В результате чего все большее число традиционных источников новостей сообщают о необоснованной информации из-за спешки быть первыми.

Аварийные службы должны иметь дело с последствиями слухов и охоты на ведьм в социальных сетях, и, наконец, у новостных потребителей есть невероятно трудная задача просеивания новостей, чтобы отделить обоснованные и заслуживающие доверия новости от слухов и необоснованных предположений.

Актуальность работы:

Во время взрыва на Бостонском марафоне, когда были пущены ложные слухи о внешности одного из террористов, а также об их количестве, что привело к неоправданным действиям со стороны гражданских лиц.

Данный случай демонстрирует, то, как легко человек реагирует на незначительные риски. Если допускать распространение слухов на глобальном уровне, то это приведет к необычайно огромным проблемам с верой в достоверность новостного потока.

Что, если бы был инструмент, который мог не только обнаруживать слухи, как они распространяются, но и предсказывать достоверность этих слухов?

Со времен взрыва было представлено несколько работ посвященных этой теме. Большое количество психологов посвятили большое количество времени выявлению структуры слуха со стороны психологического анализа взаимодействия пользователей сети.

Некоторые исследования ушли в область поиска самого первого узла, который запустил волну неверных слухов [29]. Этот путь кажется логичным, ведь по первому узлу проще оценить степень достоверности новости, которую он распространяет.

Более поздние работы решили пойти дальше и определять слух до того, как он распространился.

Среди них можно выделить несколько основных подходов к решению данной задачи:

1. Извлечение временных и причинно-следственных связей между событиями [8, 9, 20]. Основой данного метода является связывание и сопоставление сообщений, посвященных одному событию с помощью причинных и временных связей.

2. Семантический анализ сообщений. Основой данных методов является разбитие сообщения на составляющие и применение семантически-сравнительного анализа сообщений. Так же берется за факт присутствие конфликта между данными [2, 3, 21, 26].

3. Большой вклад внести работы, посвященные автоматическому поиску слухов с помощью использования технологий проектирования искусственного интеллекта. Данные работы основываясь на работах, проведенных в 1-2пп. проводят анализ сообщений с помощью классификаторов и выявляют вероятность правдивости слуха. Для данных работ важной особенностью является правильно организованная выборка данных для обучения искусственной нейронной сети [10, 24, 27].

Традиционные методы решения задачи определения вероятности подобного рода в основном направленны на составления нейронной сети для каждого конкретного случая. Такие методы, как правило, требуют выполнения следующих шагов для каждой из систем, в случаях реализации метода: поиск источников информации, семантическую обработку запросов, произвести работу с метаданными классификация, идентификация пользователей, извлечение особенностей и нейронная сеть.

Среди представленных выше моделей, в качестве наиболее полноценного можно выделить работу из Массачусетского университета под предводительством Соруша Восауи (Soroush Vosoughi). В своей работе он представил совокупность методов, которая привела к лучшим результатам. В отличии от других методов, Соруш решает не одну задачу, а комплекс подзадач, которые решают одну большую задачу.

Важным фактом являет то, что большинство данных работ вели свою деятельность в сети Twitter, что не удивительно, ведь по данным сайта internetlivestats.com ежедневно появляется более 500млн. сообщений в день. Что является хорошим показателем легкости распространения новостей в сети интернет.

Данная задача за последние 5 лет снискала огромную популярность на фоне политических конфликтов, террористических атак и провоцировании общества на совершение действий, выгодных для нарушения спокойной ситуации на социальном уровне. И из этого вытекает цель для данной задачи.

Цель:

Уменьшение негативного влияния слухов на социальную сферу общества и уменьшение вероятности дальнейшего распространения.

Для достижения указанной цели в работе поставлены следующие **задачи:**

1. Предобработка данных:
    1) Проанализировать признаки;
    2) Чистка данных;
    3) Поиск связей:
    4) Нормализация;
    5) Проанализировать признаки;
    6) Визуализировать.
2. Разбиение обработанных данных на обучающую и тестирующую выборки:
    1) Принцип разбиения выборки;
    2) Кросс-валидация;
3. Разработка модели нейронной сети;
4. Разработка приложения;
5. Разработка графического интерфейса.

Применяемые методы

1. Язык программирования - Python;
2. Twitter API;
3. Библиотеки для Python (Matplotlip, scikit-learn, numpy…);
4. Имитационное моделирование, для испытания параметров.

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящей работе применяют следующие термины с соответствующими определениями.

Социальная сеть — это социальная структура, состоящая из групп узлов, которыми являются социальные группы, личности, индивидуумы.

Twitter – Социальная сеть для публичного обмена сообщениями.

Tweet (Сообщение) – текстовое сообщение, длиной до 280 символов. Может содержать ссылки на изображение, видео и т.д., которые не влияют на длительность самого сообщения.

Ретвит (Retweet) – копия текстового сообщения другого пользователя с указанием на него ссылки.

Отметки «Нравится» (Like) - Выступает в качестве одного из показателей популярности поста среди пользователей. Так же выступает в качестве функции сохранения в избранное.

Фолловеры (followers) – Пользователи, которые «читают» другого пользователя, т.е. получают от пользователя его сообщение.

Читаемые (Followed) – набор пользователей, которых читает один пользователь.

Проверенный пользователь (verified)– пользователь, которому система Twitter предоставила значок «Проверен». Такой значок получают популярные пользователи, медийные личности и новостные страницы.

Хэштег (Hashtag) – специфичное слово отмеченного хешом (#). Данное слово выступает, словом-ассистентом, для поиска и упоминания сообщения пользователя в списке твитов обсуждающих определенную тему.

Слух – Утверждение, достоверность которого невозможно проверить в данный момент, но с возможностью определить достоверность позднее, передается от одного пользователя к другому через сети.

Деревья – структура распределения данных. Имеет главную ячейку и побочные отходящие либо в право, либо в лево. Крайние ячейки называются листья.

Сэмплирование - метод получения репрезентативной выборки пользователей социальной сети.

Диффузию твитов - анализ пути ретвита этих твитов.

Фактчекинг – проверка достоверности сведений, описанных в работах. Одним из примеров фактчекинга является рецензирование работ экспертами.

Искусственный интеллект – свойство интеллектуальных систем выполнять творческие функции, которые традиционно считаются прерогативой человека.

Нейрон – упрощенная модель биологического нейрона.

Нейронная сеть – совокупность искусственных нейронов или упрощенная модель биологических нейронных сетей.

Параметры нейронной сети – набор параметров, определяющий поведение нейронной сети при обучении и работе с данными.

Слой – одна из частей структуры нейронной сети, содержит в себе набор нейронов.

Веса – параметр, который изменяет входные данные при переходе от одного нейрона к другому.

Обучающая выборка – набор данных, которыми управляет нейронная сеть при работе.

Тестирующая выборка – набор данных, которыми выполняется проверка работы нейронной сети.

Функция активации – способ организации входных данных, для получения выходных данных, удовлетворяющих запросам.

Синапс – связь между двумя нейронами.

Итерация – Счетчик прохода одной тренировочной выборки.

Эпоха – Количество тренировок, чем больше эпоха, тем лучше нейронная сеть натренирована.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

API (application programming interface) - программный интерфейс приложения, набор особенностей при помощи которых одна программа может взаимодействовать с другой.

HTML – стандартизированный язык разметки документов во Всемирной паутине.

СМИ – Средства массовой информации

LCC – наибольший компонент связности. Поддерево содержащие наибольшее количество узлов.

ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ.

1.1. Связанные работы

Проблеме поиска и проверки слухов посвящено не так много трудов, но и среди них присутствуют различные методологии решения данной задачи.

Например, при помощи рекуррентной нейронной сети, подавая на вход не обработанные параметры полученные из API. Таким системам посвящены работы Weiling Chen, Yan Zhang, Chai Kiat Yeo, Chiew Tong Lau, Bu Sung Lee, данная группа специалистов из Китая предлагает находить слухи, но не определять являются ли полученные системой сообщений слухами, да и работает данная система только на китайской платформе Weibo [27].

Другая же команда из Китайского Гонг Конского университета, во главе с Jing Ma, Wei Gao, Prasenjit Mitra, предлагает схожую идею с рекуррентной нейронной сетью, но в тоже время предлагает обрабатывать весь набор сообщений за определенный срок. По их информации для одной обработки одного запроса необходимо 1582 часа, что не позволительно для такой важной темы как слух , который распространяется очень быстро [9].

Еще одна команда из Китайского института компьютерных технологий и Китайской академии наук включающая Zhiwei Jin, Juan Cao, Han Guo, Yongdong Zhang и Yu Wang, решила не охватывать все возможные темы, а сосредоточится на конкретной теме распространения слухов вовремя президентский выборов в США. В своей работе, зная заранее про новости, которые считаются слухами, они собрали большое количество твитов за определенный период времени и с помощью классификатора выполняли поиск аккаунтов, которые данные новости распространяли. Такая работа требует много времени и результат никак не мешает распространению слуха [9].

Несколько работ посвящены обработке лингвистических особенностей в тексте, а именно поиск «сигнальных» слов, указывающих на возможный слух. Такому методу посвящены работы группы ученых из Мичиганского

университета в лице Zhe Zhao, Paul Resnick и Qiaozhu Mei [29]. Развивают эту идею Alton Y. K. Chua и Snehasish Banerjee, которые предложили использовать не только сигналы, но и нейронную сеть с различными параметрами, описывающими текст, такие как стиль написанного текста, количество символов и другие [2].

Наследует данную идею Tetsuro Takahashi, который в своей работе использует систему фильтрации твитов используя различные комбинации, включая и лингвистические особенности [26].

Следующий метод предлагает поиск слухов при помощи набора параметров и нейронной сети. Эти параметры делятся на 3 составляющие:

1) Специфичные для сети Twitter маркеры, такие как ответы, ретвиты, наличие ссылок и хэштеги;
2) Основанные на контенте твита исследуемого при помощи n-gramm;
3) Внешние факторы;

Авторами данной концепции являются Sardar Hamidian и Mona Diab из Университета Джорджа Вашингтона [3].

Самую крупную и самую важную долю в исследование внес Soroush Vosoughi из Массачусетского института технологий. В своей работе он описал наиболее комплексное решение данной проблемы. Решение во многом задействует представленные выше концепты. Работа делится на 2 составляющие:

1) Автоматический поиск слухов по запросу;
2) Определение является ли сообщение слухом или нет.

Все это осуществляется при помощи комбинации работы нейронной сети, классификаторов, лингвистических особенностей текста, временных факторов и особенностей поведения пользователя [24]. На данную работу и опирается диссертация.

1.2. Использование социальных сетей для сбора данных

Социальная сеть — это социальная структура, состоящая из групп узлов, которыми являются социальные группы, личности, индивидуумы. Одна из обычных черт социальных сетей — это система «друзей» и «групп». Социальная сеть облегчает создание персонального профиля и виртуальных взаимоотношений.

Социальные сети используются для поиска людей со схожими интересами и объектов этих интересов. В социальных сетях обычно используется народная классификация, практика совместной категоризации информации [30].

Twitter — микроблогавая социальная сеть. Используется как новостная социальная сеть друзей и поклонников. Твиттер является микроблогом и средой для общения друзей с помощью коротких сообщений. Многие пользователи ведут твиттер-блог через мобильные телефоны, смартфоны, карманные компьютеры, используя мобильную версию социального сайта.

Для получения актуальных данных необходимо обеспечить оперативный сбор информации и ее хранение в базе данных в унифицированном формате [33].

Для того чтобы начать процесс сбора данных, нужно удостоверится что эти данные существуют, и узнать о том, если ли доступ к ним. Но получение данных, даже из готовой базы не всегда является простой задачей. Каждая организация заботится о свой безопасности и конфиденциальности информации, поэтому сотрудник, который занимается работой с данными, должен следить за тем, чтобы не нарушать закон [32].

Вместе с тем, при работе с социальными данными нужно принимать во внимание такие факторы, как нестабильность качества пользовательского контента (спам и ложные аккаунты), проблемы с обеспечением приватности личных данных пользователей при хранении и обработке, а также частые

обновления пользовательской модели и функционала. Всё это требует постоянного совершенствования алгоритмов решения различных аналитических и бизнес-задач.

Таким образом социальные сети являются хорошим средством для получения данных посредством обработки информации о пользователях.

1.3. Сбор данных

Веб-интерфейсы социальных сетей являются источниками данных в режиме реального времени и предназначены для просмотра и взаимодействия со страницами социальной сети в веб-браузере либо для использования данных пользователей специализированными приложениями.

Благодаря почти постоянному обновлению новых сообщений и общедоступного API, Twitter может стать полезным источником для сбора данных, которые будут использоваться для изучения ряда проблем, связанных с обработкой естественного языка и распространением информации [30].

Поскольку сценарии использования интерфейсов социальных сетей не предполагают автоматического сбора данных множества пользователей, то возникает ряд проблем:

1. приватность данных - зачастую доступ к данным пользователей разрешён только для зарегистрированных и авторизованных участников сети, что требует поддержки эмуляции пользовательской сессии с помощью специальных учётных записей (аккаунтов);

2. слабая структурированность данных - во многих случаях программные интерфейсы (API) социальных сетей имеют ограниченный функционал, что требует поддержки получения с помощью пользовательского веб-интерфейса статических копий HTML-страниц, корректной обработки их динамической части (включая исполнение асинхронных запросов к серверу социальной сети), извлечения нужных данных с помощью алгоритма и/или

шаблона и построения их структурированного представления, удобного для дальнейшей автоматической обработки;

3. ограничения доступа и блокировки - с целью предотвращения несанкционированного автоматического сбора данных и ограничения нагрузки на инфраструктуру сервиса социальной сети владельцы сервисов зачастую вводят явные или скрытые ограничения на допустимое количество запросов от одного пользовательского аккаунта и/или IP-адреса в единицу времени, что требует учёта количества посылаемых запросов, а также поддержки динамической ротации используемых для сбора данных пользовательских аккаунтов и IP-адресов;

4. размерность данных обуславливает необходимость в параллельном методе сбора данных, а также в методах получения репрезентативной выборки пользователей социальной сети (сэмплирование) [31].

Существует несколько способов поиска информации для сбора данных. Как говорилось ранее существуют работы, которые для поиска используют «сигналы». Данными сигналами может быть предложение, словосочетание, слово или их набор.

В некоторых существующих работах в качестве сигнала используются исправления, сделанные авторитетными источниками или пользователями социальных сетей. Например, Такахаши и Игата [26] отследили ключевое слово «ложные слухи».

В работах таких авторов как Kwon [13], Friggeri [5] и Alton Y. K. Chua [2] отслеживали суждения, сделанные на сайтах по разоблачению слухов, таких как Snopes.com. Данный сайт определяет являются ли новости распространением слухов, а также определяет уровень слуха, т.е. является слухом, частично является и не является. Российским аналогом такого сайта является https://noodleremover.news и фактчекинг от google.

1.3.1. Методы сбора и обработки данных

Первый модуль системы состоит из двух основных этапов, т.е. получение и очистка твитов, а также предварительная обработка твитов.

Существуют несколько методов получения данных из сети Twitter:

1) Через API (Application Programming Interface);
2) Существующие дополнительные инструменты;
3) Специализированные компании по продаже данных;
4) Сайты распространяющие выборки данных.

На этапе выборки и очистки твиты выбираются в соответствии с некоторыми критериями поиска (например, ключевые слова, время и дата публикации, место размещения, хэштеги). Дополнительным критерием поиска было наличие ссылки ведущей на новость, являющуюся слухом. Хотя можно прибегнуть к настраиваемым инструментам, разработанным для этой цели, нашим главным требованием был полный охват соответствующих твитов (и это не гарантируется использованием простых API Twitter).

Загруженный набор необработанных твитов уменьшен с целью отбрасывания:

• дубликатов твитов, т. е. твитов с одинаковыми идентификаторами;

• твиты, написанные на других языках, кроме целевого (русский): это может произойти из-за наличия ключевых слов / хэштегов с одинаковым написанием на разных языках.

Что касается ретвитов (т. е. твиты других пользователей просто пересылаются), то есть смысл сохранить их в наборе данных, так как ретвитинг действий, в этом контексте, является способом поддержки / обмена такого же мнения другого пользователя.

Предварительная обработка. На этом втором шаге твиты предварительно обрабатываются с применением фильтра регулярных

выражений, чтобы извлечь только текст каждого твита и удалить всю ненужную метаинформацию. Фактически, каждый извлеченный необработанный твит содержит идентификатор твита, идентификатор пользователя, временную метку, местоположение (если предусмотрено), флаг ретвита и содержание твита.

Содержимое твита может включать в себя текст пользователя, хэштеги, ссылки и упоминания. С использованием фильтра регулярных выражений идентификаторы твита, идентификаторы пользователя, местоположения и флаг ретвита отбрасываются. Временная метка временно отбрасывается для целей разработки интеллектуального анализа текста, но она будет использоваться для анализа динамики распространения.

Из содержимого твита отбрасываем: ссылки, упоминания, цифры и специальные символы (например, знаки препинания, скобки, косые черты, кавычки и т. д.). Хештеги не отбрасываются полностью, вместо этого они сводятся к словам (исключая символ хеша (#)), чтобы не потерять соответствующую информацию. На самом деле, обычный способ написания пользователей Twitter — это использование хэштегов в предложениях вместо обычных слов.

Наконец, к текстам применяется операция сворачивания регистра, чтобы преобразовать все символы в строчную форму. Следовательно, каждый твит представлен в виде последовательности символов. Обозначим j-й твит множества как твит j = 1 ,…, N, где N - общее количество твитов, рассмотренных на последующих шагах [4, 17].

1.3.2. Доступные инструменты для сбора и анализа данных

Для сбора и анализа данных в социальной сети Twitter, если не брать в расчет официальный API, было найдено в общей сложности 9 инструментов. Рассмотрим каждый инструмент относительно необходимых возможностей для реализации:

TAGS: Данный инструмент выполнен с помощью google sheets. Имея довольно примитивный интерфейс приложение предоставляет возможность быстрого сбора данных из сети Twitter, но только на английском языке. Приложение способно собрать вплоть до 3000 твитов за раз и делает это с довольно быстро. Но не собирает важные данные о твитах, которые необходимы для поиска слуха. Требует дополнительных обработок полученных данных.

Mozdeh: Данное приложение является мощным инструментом для анализа больших данных. Присутствует поиск, получение твитов, данных о твитах, а также анализ текста. Имеет возможность работы с несколькими языками, включая русский. Одной из сильных сторон является определение эмоционального оттенка сообщения. Имеет интерфейс и мощную графическую визуализацию данных. Имеет ограничения наследованные от API Twitter, а именно данные можно получить за последние 7 дней и не более 72000 записей в час. Умеет работать в автономном режиме и собирать данные без остановки.

Chorus: Позволяет обработать только существующую выборку твитов. Имеет веб интерфейс, разделенный на две программы. Первая позволяет отправлять запросы по API сети Twitter и получать данные, а вторая позволяет анализировать полученные данные. Имеет те же ограничения, что и API. Работает только для английского языка.

Netlytic: Данное приложение имеет веб интерфейс и позволяет обработать как уже существующую выборку твитов, так и составить новую. Основной особенностью данного приложения является поиск ключевых слов в тексте, категоризация твитов и анализ социальных связей между пользователями. Наследует те же проблемы, что и приложения выше.

Twython: Простая библиотека для языка Python, которая поверх официальной библиотеки Twitter накладывает дополнительные возможности для анализа и выгрузки данных.

KNIME: Инструмент для анализа уже готовой выборки, сбор данных отсутствует. Довольно мощный инструмент для анализа данных. Имеет интерфейс, визуализации графиков. Имеет возможности для рисования структур процессов.

NodeXL: Данное приложения является модулем для Excel. Построение графа социальных связей по уже готовой выборке. Работает только с готовыми данными и является в основном библиотекой для поиска связей между пользователями в социальных сетях.

Visibrain: Коммерческое приложение имеющее демо версию. Бесплатной версии не предусмотрено.

Проведя исследование доступных инструментов, можно сделать вывод, что большая часть либо не способна собирать данные, либо имеет ограничения, связанные с официальными ограничениями на API Twitter. Рассмотрев данные приложение было решено использовать приложение Mozdeh позволяющее наиболее максимально получить данные русском языке. Также полезным будет получение оценки эмоционального оттенка полученного сообщения.

К сожалению, из-за ограничений данного приложения следует провести дополнительные действия для получения дополнительных метрик для анализа.

Таким образом сбор данных осуществляется при помощи комбинировании специализированного инструмента и официальной библиотеки Twitter. Чтобы собирать данные необходимо определить значимые данные.

1.4. Определение значимых данных и получение репрезентативной выборки

Существует несколько типов признаков. У каждого типа свои особенности: как нужно их обрабатывать и учитывать в алгоритмах машинного обучения.

- ***Бинарные признаки***.

Могут принимать всего два значения:

Если ответ на некоторый вопрос - «да», то признак полагается равным 1, если ответ на вопрос «нет» - равным 0.

- ***Вещественные признаки***.

Возможные значения – действительные числа.

- ***Категориальные признаки***.

Неупорядоченное множество. Допускается сравнение значений только на совпадение (но не на >, <). Требуется применение специальных методов работы с такими признаками.

- ***Порядковые признаки***.

Упорядоченное множество. Отличие от вещественных признаков: «расстояние» между значениями признака не имеет смысла.

- ***Множествозначные признаки (set-valued)***.

Множество всех подмножеств некоторого множества. Каждое значение признака – одно из этих подмножеств.

Для достижения поставленной цели необходимы данные, которые можно получить используя API сети Twitter, который поставляет набор параметров для каждого твита, включающий 54 наименования [12]. Так как для поставленных задач нет нужны использовать их все необходимо

определить набор параметров включащий только необходимые. Перечислим необходимый набор параметров в таблице ниже:

Таблица 1. Необходимый набор данных

Название столбца	Тип признака	Описание
ID	Integer	Номерной идентификатор пользователя в сети Twitter. Целое число.
Verified	Bool	Булево число, обозначающее верификацию пользователя. 1 – проверен, 0 иначе.
Folowers_Count	Integer	Количество подписчиков пользователя. Целое число.
Friends_Count	Integer	Количество пользователей, на которые подписан конкретный пользователь. Целое число.
Favourites_Count	Integer	Количество отметок «Нравится». Целое число.
Created_at	Time	Дата и время, когда был зарегистрирован аккаунт пользователя. Хранится в следующем виде: *ДеньНедели Месяц День hh:mm:ss +0000 yyyy*
Statuses_Count	Integer	Количество твитов (Сообщений) Пользователя. Целое число.
Retweet_count	Integer	Количество ретвитов среди всех сообщений, которые сделал пользователь. Обозначаются @RT + text
Tweet_ID	Integer	Номерной идентификатор твита в сети Twitter. Целое число.

Tweet_text	String	Текст твита, который может включать

В основе программы лежит распознавание «сигналов» поступающих вместе с твитом.

Ответ, повторный твит, идентификатор пользователя. Ответы и ретвиты в микроблогах являются факторами, позволяющими судить о надежности пользователя, когда речь заходит о передаче информации [12]. Например, пользователь А с большей вероятностью отправит слух, чем пользователь В, если у пользователя А есть история ретвитов или ответов пользователю С, у которого также есть история распространения слухов.

Изучение правдоподобности пользователей является дорогой и почти невозможной задачей, но выполнимой, когда мы хотим исследовать только конкретную историю. Например, знание ограниченного числа пользователей, у которых есть история публикации слухов, может быть подсказкой для обнаружения большого количества пользователей, которые следят, ретвитят или отвечают на эти твиты.

Набор значимых данных описанный в таблице 1 является минимальным необходимым набором для получения выборки данных для обучения и тестирования нейронной сети. Собранный набор данных определит модель нейронной сети для задачи проверки твита на слух.

1.5. Обозначение задач

Лингвистические особенности фиксируют характеристики текста твитов в слухах. Было найдено, что в общей сложности 5 лингвистических особенности вносят существенный вклад в результаты наших моделей [26]. В порядке убывания вклада эти характеристики: соотношение твитов, содержащих вульгарные слова, наличие смайликов, сокращения и аббревиатуры, и сложность твитов, теперь опишем каждую из этих функций подробно.

Пользовательские функции отражают характеристики пользователей, участвующих в распространении слухов. Было обнаружено, что всего 6 пользовательских функций внесли незначительный вклад в результат наших моделей. В порядке убывания вклада эти характеристики: противоречивость, оригинальность, достоверность, влияние, роль и вовлеченность.

Особенности распространения отражают динамику временной диффузии слухов. Было обнаружено, что в общей сложности 7 функций распространения вносят значительный вклад в результат наших моделей. В порядке убывания вклада эти характеристики: доля диффузии от низкой к высокой, доля узлов в наибольшем связанном компоненте (LCC), отношение средней глубины к ширине, отношение новых пользователей, отношение исходных твитов, доля твитов, содержащих внешние ссылки и доля изолированных узлов. Все эти особенности получены из графика распространения слухов. Прежде чем мы подробно опишем эти особенности, нам нужно объяснить, как был создан диффузионный граф.

1.6. Модель

1.6.1. Определение модели

Имеется:

- множество объектов (ситуаций)
- множество возможных ответов (откликов, реакций).

Между ответами и объектами существует некоторая зависимость, но она неизвестна.

1.6.2. Топология модели

Учитывая данные $\vec{x} = \{x_1, \dots, x_n\}$ из n особенностей, Наивный Байес предсказывает класс C_k для $\vec{x}$ в соответствии с вероятностью.

$$p(C_k|\vec{x}) = p(C_k|x_1, \dots, x_n), \text{ где } k = 1 \dots, K \quad (1)$$

Использую теорему Байеса, это можно учесть, как:

$$p(\mathrm{C}_k|\vec{x}) = \frac{(p(\mathrm{C}_k|\vec{x})p(\mathrm{C}_k))}{P(\vec{x})} = \frac{p(x_1, \dots, x_n|\mathrm{C}_k)p(\mathrm{C}_k)}{p(x_1,\dots,x_n)} \quad (2)$$

Используя правило цепи, фактор $p(x_1, \dots, x_n|\mathrm{C}_k)$ в числителе может быть дополнительно разложен как:

$$p(x_1, \dots, x_n|\mathrm{C}_k) = \quad (3)$$
$$p(x_1|x_2, \dots\ x_n, \mathrm{C}_k)p(x_1|x_2, \dots\ x_n, \mathrm{C}_k)\dots\ p(x_{n-1}|n, \mathrm{C}_k)p(x_n|\mathrm{C}_k)$$

В этот момент вводится в действие «наивное» условие условной независимости. В частности, наивные байесовские модели предполагают, что функция x_i не зависит от функции x_j для данного класса C_k. Используя предыдущее разложение, это можно сформулировать как:

$$p(x_i|x_{i+1}, \dots, x_n|\mathrm{C}_k) = p(x_i|\mathrm{C}_k) => \ p(x_1, \dots, x_n|\mathrm{C}_k) \ = \quad (4)$$
$$\prod_{i=1}^{n} p(x_i|\mathrm{C}_k)$$

Таким образом

$$p(\mathrm{C}_k|x_1, \dots, x_n) \ \propto \ p(\mathrm{C}_k|x_1, \dots, x_n)$$
$$\propto p(\mathrm{C}_k)p(x_1, \dots, x_n|\mathrm{C}_k) \quad (5)$$
$$\propto p(\mathrm{C}_k)p(x_1|\mathrm{C}_k)p(x_2|\mathrm{C}_k) \dots p(x_n|\mathrm{C}_k)$$
$$\propto p(\mathrm{C}_k) \prod_{i=1}^{n} p(x_i|\mathrm{C}_k)$$

Практически говоря, класс условных признаков вероятностей $p(x_i|\mathrm{C}_k)$ обычно моделируются с использованием одного и того же класса распределения вероятностей, такого как биномиальное распределение или распределение Гаусса. В последнем случае условное распределение классов известно, как произведение гауссианов и обладает некоторыми удобными вычислительными свойствами, которые упрощают уравнения оптимизации для обучения параметров.

Наивный Байес дает вероятность того, что точка данных $\vec{x}$, принадлежащая классу C_k, пропорциональна простому произведению n+1 факторов (класс априорный $p(\mathrm{C}_k)$ плюс n вероятности условного признака $p(x_i|\mathrm{C}_k)$). Поскольку классификация включает присвоение класса C_k точке данных, для которой значение $p(x_i|\mathrm{C}_k)$ является наибольшим, этот

пропорциональный продукт может использоваться для определения наиболее вероятного присвоения класса. В частности,

$$p(\mathrm{C}_a)\prod_{i=1}^{n} p(x_i|\mathrm{C}_a) > p(\mathrm{C}_b)\prod_{i=1}^{n} p(x_i|\mathrm{C}_b) \Rightarrow p(\mathrm{C}_a|x_1,\dots,x_n) > p(\mathrm{C}_b|x_1,\dots,x_n) \quad (6)$$

Таким образом, наиболее вероятное назначение класса для точки данных $\vec{x} = x_1,\dots,x_n$ можно найти, рассчитав $p(\mathrm{C}_k)\prod_{i=1}^{n} p(x_i|\mathrm{C}_k)$, где k = 1 ,…, K и присвоение $\vec{x}$ класса C_k, для которого это значение является наибольшим. В математической записи это определяется:

$$\hat{C} = argmax_{\,k\in\{1,\dots,K\}} p(\mathrm{C}_k)\prod_{i=1}^{n} p(x_i|\mathrm{C}_k), \quad (7)$$

где $\hat{C}$ это оценочный класс $\vec{x}$ с учетом его особенностей $x_1,\dots,x_n$.

1.6.3. Метод обучения

Различают 2 основных направления обучения нейронных сетей:

Обучение с учителем. Известна конечная совокупность прецедентов – пар вида (объект, ответ), называемая обучающей выборкой. На основе этих данных требуется построить модель, которая будет выдавать ответы для новых объектов (не включенных в обучающую выборку). Обучение с учителем использую в таких задачах как бинарная классификация, многоклассовая классификация, регрессия и ранжирование [33].

Обучение без учителя. Есть только совокупность объектов; правильные ответы неизвестны. Такие методы, как правило, труднее интерпретировать и оценивать. Обучение без учителя применяется для кластеризации, задач визуализации и поиск аномалий.

Для нахождения оптимальных весов w могут использоваться различные методы оптимизации, в частности, метод градиентного спуска (для дифференцируемой функции ошибки), метод стохастического градиентного спуска и др.

Один из наиболее популярных алгоритмов – алгоритм обратного распространения ошибки (back propagation, backprop***)***.

Для ***i*** -го нейрона первого скрытого слоя (***i*** = 1, 2, …, n_1) на каждом объекте ***x*** обучающей выборки выполняется:

- вычисление взвешенной суммы входных сигналов нейрона

$$u_i^{(1)} = \langle w_i^{(1)}, x \rangle = \sum_{j=1}^{d+1} w_{ij}^{(1)} \cdot x^j \quad (8)$$

- вычисление выходного сигнала нейрона

$$\gamma_i^{(1)} = \sigma^{(1)}(u_i^{(1)}) \quad (9)$$

Для ***i*** -го нейрона ***k***-го скрытого слоя, ***k*** = 2, …, ***K*** на каждом объекте ***x*** обучающей выборки выполняется:

- вычисление взвешенной суммы входных сигналов нейрона

$$u_i^{(k)} = \langle w_i^{(k)}, \gamma^{(k-1)} \rangle = \sum_{j=1}^{d+1} w_{ij}^{(1)} \cdot \gamma_i^{(k-1)} \quad (10)$$

- вычисление выходного сигнала нейрона

$$\gamma_i^{(k)} = \sigma^{(k)}(u_i^{(k)}), i = 1,2,\dots,n_k \quad (11)$$

где n_k – число нейронов на ***k***-м слое.

Для определения весов, минимизирующих значение ошибки

$$\varphi(w, x) = z^{(k)}\left(u^{(k)}\right) - y \quad (12)$$

Используется градиентный спуск. При этом градиент функции ***φ*** (***w***, ***x)*** вычисляется по алгоритму ***обратного распространения***: при вычислении компонентов градиента, относящихся к ***k***-му слою, используются значения выходов нейронов (***k***+1)-го слоя.

Компоненты градиента могут быть найдены в соответствии с равенствами:

$$\frac{\partial \varphi}{\partial w_{ij}^{(k)}} = \left(\sum_{p=1}^{n_{k+1}} \frac{\partial \varphi}{\partial w_p^{(k+1)}} \cdot \frac{d\sigma^{(k)}(u_p^{(k+1)})}{du_p^{(k+1)}} \cdot w_{ip}^{(k+1)}\right) \frac{d\sigma^{(k)}(u_i^{(k)})}{du_i^{(k+1)}} \cdot \gamma_j^{(k-1)} \quad (13)$$

В случае, если функции активации нейронов всех слоев являются сигмоидами $\boldsymbol{\sigma}$,

$$\frac{d\sigma^{(k)}\left(u_i^{(k)}\right)}{du_i^{(k+1)}} = \frac{d\sigma\left(u_i^{(k)}\right)}{du_i^{(k+1)}} = \beta \cdot \gamma^{(k)}\left(1 - \gamma_i^{(k)}\right) \quad (14)$$

Формула нахождения $\frac{\partial \varphi}{\partial w_{ij}^{(k)}}$ позволяет определить компоненты градиента для весов всех слоев, кроме последнего.

Компоненты градиента для весов последнего слоя (при k = K) определяются в соответствии с

$$\frac{\partial \varphi}{\partial w_{ij}^{K}} = (\gamma^K - y) \cdot \gamma_j^{K-1} \cdot \frac{d\sigma^{(k)}\left(u^{(K)}\right)}{du^{(K)}} \quad (15)$$

Преимущества алгоритма обратного распространения ошибки.

a. Градиент вычисляется за время, сравнимое с временем вычисления сети.
b. Алгоритм подходит для многих дифференцируемых функций активации.
c. В процессе обучения не обязательно использовать всю выборку.

Недостатки алгоритма обратного распространения ошибки.

a. Возможна медленная сходимость к оптимуму.
b. Полученное решение может оказаться локальным, а не глобальным минимумом.
c. Возможно переобучение сети.

Последовательность действий при обучении многослойного персептрона.

1. Начало цикла по эпохам.
   1.1. Случайное разбиение имеющегося набора данных на подмножества.
   1.2. Начало цикла по подмножествам.

1.2.1. Для каждого подмножества из текущего разбиения:

1.2.2. вычисление выходов сети (предсказаний) для примеров подмножества;

1.2.3. вычисление градиента функции ошибки по всем весам (включая смещение) на примерах подмножества;

1.2.4. обновление весов (в направлении антиградиента пропорционально заданному шагу).

1.3. Конец цикла по подмножествам.

1.4. Вычисление значений функции ошибки на всех имеющихся данных для сети с обновленными весами.

1.5. Если значение ошибки уменьшилось менее, чем на заданную величину или достигнуто максимальное значение количества эпох, то – выход из цикла по эпохам.

2. Конец цикла по эпохам

Описанная выше модель выполняет необходимый набор функций для работы информационной системы по определению слуха, а метод обучения поможет добиться лучшего качества работы модели [14]. Качество работы модели необходимо проверять при помощи комплекса оценок.

1.7. Проверка модели

Самый простой способ оценки качества алгоритма - разбить выборку на две части:

- первая часть - для обучения алгоритма,
- вторая (тестовая выборка) - для оценки его качества (доли ошибок в задаче классификации, среднеквадратичной ошибки (MSE) в задаче регрессии и т. п.).

Преимущество такого подхода: алгоритм обучается только один раз (меньше затраты ресурсов).

Недостаток: результат сильно зависит от того, как именно было произведено разбиение.

Существуют возможности позволяющие оптимизировать метод разбиения выборки, среди этих методов:

a. построить (случайным образом) ***n*** различных разбиений исходной выборки на 2 части;
b. для каждого из ***n*** разбиений найти оценку качества алгоритма;
c. в качестве итоговой оценки качества использовать усредненное по всем разбиениям значение.

Но нет вероятности того, что каждый объект выборки попадет обучающую выборку хотя бы раз. Поэтому более систематический метод — это проведение кросс-валидации.

Суть кросс-валидации:

a. исходная выборка делится на ***k*** блоков примерно одного размера;
b. каждый из этих блоков по очереди используется в качестве тестового, а все остальные - в качестве обучающей выборки;
c. для каждого из ***k*** тестовых блоков определяется показатель качества алгоритма;
d. результат усреднения этих ***k*** показателей - оценка качества по кросс-валидации.

Так же для кросс-валидации существуют варианты разбиения на блоки, рассмотрим самые популярные:

KFold - разбиение набора данных на k блоков (folds), каждый из которых один раз участвует в тестировании и k -1 раз – в обучении. По умолчанию объекты помещаются в блоки по порядку (без перемешивания).

StratifiedKFold - при формировании блоков сохраняется соотношение классов в обучающих и тестовых выборках (для задач классификации).

ShuffleSplit - каждое новое разбиение строится независимо от предыдущих (объекты, использованные для обучения и теста на предыдущей итерации, возвращаются в исходный набор и перемешиваются) каждый объект может неоднократно участвовать как в обучении так и в тестировании.

StratifiedShuffleSplit – при формировании блоков сохраняется соотношение классов в обучающих и тестовых выборках (для задач классификации).

LeaveOneOut – каждый объект используется ровно один раз в качестве тестовой выборки (состоящей из одного объекта); остальные объекты образуют обучающую выборку.

Рассмотрим другие варианты оценки модели.

Естественная оценка ошибки классификатора – доля неправильных ответов:

$$\frac{1}{k}\sum_{i=1}^{k}[a(x_i) \neq y_i] \quad (16)$$

Но в задачах классификации принято выбирать метрику так, чтобы ее нужно было максимизировать (в отличие от регрессии, где ошибка минимизировалась).

Простая и широко используемая метрика:

$$accuracy(a, X) = \frac{1}{k}\sum_{i=1}^{k}[a(x_i) = y_i] \quad (17)$$

Однако данная метрика имеет свои ограничения, а именно метрика не позволяет адекватно оценить качество алгоритма при несбалансированных данных.

Еще одним методом является расчет матрицы ошибок.

Таблица 2. Матрица ошибок

	y = +1	y = -1
a(x) = +1	True possible (TP)	False Positive (FP)
a(x) = -1	False Negative (FN)	True Negative (TN)

Рассмотрим варианты попаданий данных в ячейки:

a. Если результат работы алгоритма на объекте x a(x) = +1, то говорят, что алгоритм сработал на объекте x.
b. Если при этом объект x действительно относится к классу +1, то имеет место верное срабатывание (True Positive -TP);
c. Если же объект x относится к классу -1, то имеет место ложное срабатывание (False Positive - FP).
d. Если результат работы алгоритма на объекте x a(x) = -1, то говорят, что алгоритм пропускает объект x.
e. Если при этом объект x действительно относится к классу -1, то имеет место истинный пропуск (True Negative - TN);
f. Если же объект x относится к классу +1, то имеет место ложный пропуск (False Negative - FN).

Для оценки алгоритма были придуманы характеристики, основанные на матрице ошибок.

Точность:

$$presicion(a, X) = \frac{TP}{TP+FP} \quad (18)$$

Характеризует степень доверия к алгоритму, когда он срабатывает (доля истинных срабатываний в общем числе срабатываний алгоритма).

Полнота:

$$recall(a, X) = \frac{TP}{TP + FN} \quad (19)$$

Доля объектов класса +1, на которых алгоритм срабатывает (т. е. которые классификатор правильно распознает).

Во многих случаях требуется не условная оптимизация одного из показателей точности или полноты, а высокие оценки обоих показателей одновременно.

Самый примитивный способ это среднеарифметического этих характеристик:

$$A = \frac{1}{2}(precision + recall) \quad (20)$$

Но он далеко не всегда позволяет адекватно оценить качество алгоритма.

Таким же образом можно использовать минимум, для того чтобы разумный алгоритм получал более высокую оценку.

$$M = min\{precision, recall\} \quad (21)$$

Но существует вариант, при котором алгоритмы разного качества получают одинаковую оценку.

Самым надежным вариантом является использованием F-меры.

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad (22)$$

Данный способ слаживает метрику M.

Для оценки принадлежности к классу необходимо использовать ROC кривые. Данные кривые строятся в системе координат с следующими долями:

По оси абсцисс используется доля ложных срабатываний FPR:

$$FPR = \frac{FP}{FP+TN} \quad (23)$$

По оси ординат используется доля истинных срабатываний TPR:

$$TPR = \frac{TP}{TP+FN} \quad (24)$$

Начальной точкой выступает (0,0), а конечной (1,1). Для идеального классификатора кривая проходит через точку (0,1), чем ближе кривая к данной точке, тем лучше качество классификации.

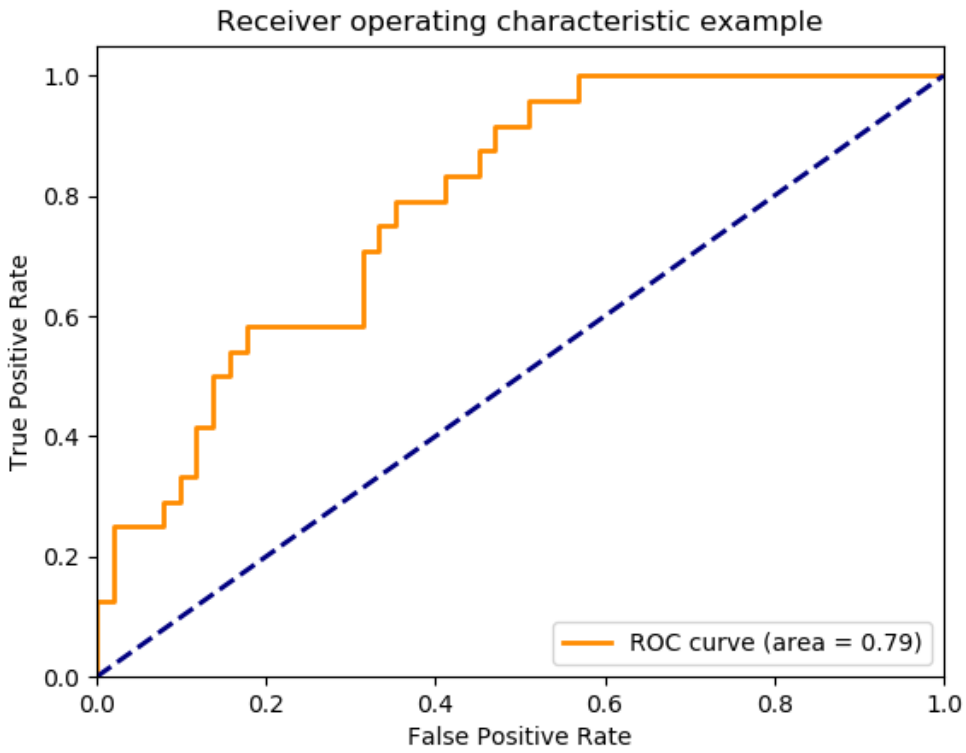

Рисунок 1. Пример ROC кривой

Кривые ROC обычно используются в двоичной классификации для изучения результатов работы классификатора. Чтобы расширить кривую ROC и область ROC для классификации нескольких классов или нескольких меток, необходимо преобразовать выходные данные в двоичную форму. Для каждой метки можно нарисовать одну кривую ROC, но можно также нарисовать кривую ROC, рассматривая каждый элемент матрицы индикатора метки как двоичное предсказание (микроусреднение).

Другой оценочной мерой для мультиклассовой классификации является усреднение по макросам, которое придает равный вес классификации каждой метки.

Описанные выше метрики позволяют оценить качество работы модели для собранных данных. Оценка позволяет всесторонне провести анализ работы модели нейронной сети [11].

ГЛАВА 2. МЕТОДОЛОГИИ И СРЕДСТВА РАЗРАБОТКИ

2.1. Лингвистические особенности текста

В качестве **входных** данных выступают следующие параметры и списки:

1) Список вульгарных слов;
2) Список смайлов [28];
3) Список сокращений и аббревиатур;
4) Параметр твита: Tweet_text, т.е. текст сообщения, включающий ссылки, смайлики, специальные символы и т.д.

В качестве **выходных** данных получаем обработанные с помощью функций.

Наличие вульгарных слов. Для получения данного параметра была собрана коллекция из более 750 слов. Значение параметра принимает 1 если в тексте было найдено вульгарное слово, 0 иначе;

Алгоритм на псевдокоде:

1. **Procedure** Vulgar_Find(текст сообщения):
2. **Вход:** текст сообщения в формате string;
3. **Выход:** булево значение 1, в случае если поиск удался, 0 иначе;
4. Загрузить заранее подготовленные данные;
5. A: =0;
6. **ДЛЯ** каждого слова в списке:
7. **ЕСЛИ** слово находится в тексте сообщения:
8. A: =1;
9. **ЕСЛИ A=1:**
10. **Возвращаем значение False**
11. **ИНАЧЕ:**
12. **Возвращаем значение True**

Проверка на наличие аббревиатур и сокращений осуществляется по такому же принципу, что и поиск вульгарных слов.

1. **Procedure** Abr_Find(текст сообщения):
2. **Вход:** текст сообщения в формате string;
3. **Выход:** булевое значение 1, в случае если поиск удался, 0 иначе;
4. Загрузить заранее подготовленные данные;
5. A: =0;
6. **FOR** каждое слово в списке:
7. **ЕСЛИ** слово находится в тексте сообщения:
8. A: =1;
9. **ЕСЛИ A=1:**
10. **Возвращаем значение False**
11. **ИНАЧЕ:**
12. **Возвращаем значение True**

Еще одна прагматическая реплика — это Смайлик. Исследования по моделированию и анализу микроблогов, которые явно используют смайлик в качестве особенности, показывают его влияние на классификацию [1]. Список популярных эмоций описан в Википедии [28].

Наличие смайликов в тексте. Работает по такому же принципу, что и негативный оттенок, но принимает значения от -3 до 3;

Алгоритм на псевдокоде:

Procedure Emoji_Find (текст сообщения):

1. **Вход:** текст сообщения в формате string;
2. **Выход:** булевое значение 1, в случае если поиск удался, 0 иначе;
3. Загрузить заранее подготовленные данные;
4. A: =0;
5. **FOR** каждый смайлик в списке:
6. **ЕСЛИ** смайлик находится в тексте сообщения:

7. A: =1;
8. **ЕСЛИ A=1:**
9. **Возвращаем значение False**
10. **ИНАЧЕ:**
11. **Возвращаем значение True**

Средняя сложность слова в твите оценивается путем подсчета количества символов в каждом слове в твите и деления на общее количество слов в этом твите.

$$WC(t) = \frac{\sum_{i=0}^{NW(t)} NC(w_i)}{NW(t)} \quad (25)$$

Уравнение (25) показывает, как это делается. Здесь WC(t) относится к показателю сложности слова в твите t, NW(t) подсчитывает количество слов в твите t, а NC(w) подсчитывает количество символов в слове w.

Алгоритм на псевдокоде:

1. **Procedure** Wordcount (текст сообщения):
2. **Вход:** текст сообщения в формате string;
3. **Выход:** среднее количество символов в слове сообщения;
4. Разделить текст сообщения на список слов;
5. Очистить от специальных слов системы Twitter;
6. **FOR** каждое слово в списке:
7. Подсчет символов в каждом слове;
8. Получаем количество слов в тексте от размера массива;
9. **RETURN** Количество символов/Количество слов

2.1.1. Негативный оттенок

Для вычисления негативного оттенка сообщения была составлена коллекция самых популярных слов на русском. Каждому слову приписано значение от -5 до 5, где слова с максимальным негативным оттенком принимают значение -5, а с максимально позитивным принимают значение 5.

По каждому сообщению, проходит поиск данных слов и если слов несколько, то берется их совокупность, а если слов не найдено, то приписывается значение 0.

Поиск осуществляется при помощи регулярных выражений поэтому в коллекции присутствуют слова, содержащие символ «*» внутри или в конце слова. Данный символ означает, что заместо этого символа можно поставить 0 или более символов. Это необходимо, чтобы охватить максимальный набор слов, не собирая их в одну коллекцию.

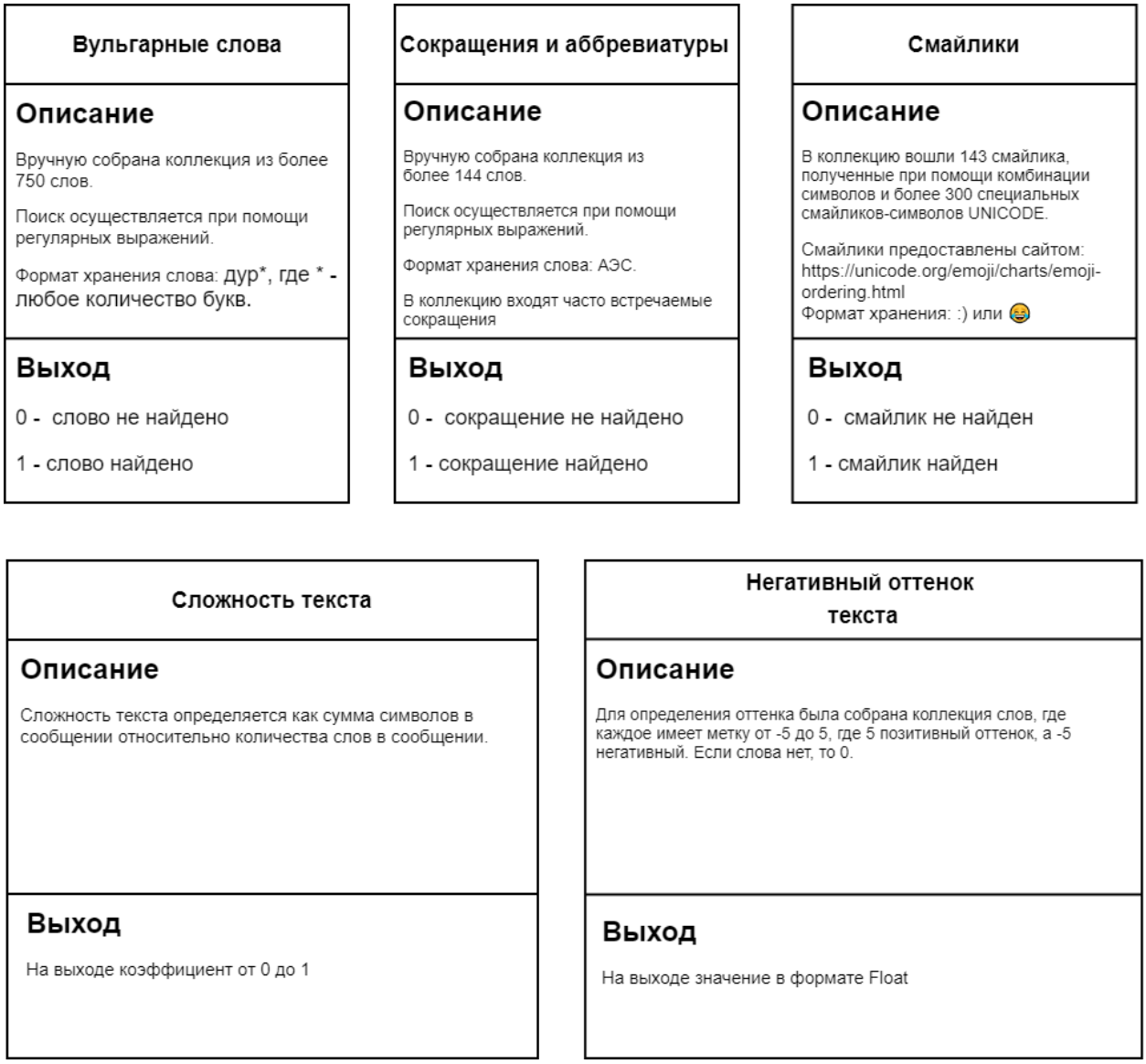

Рисунок 2. Краткое представление метрик лингвистического разбора

В итоге мы имеем набор из 5 метрик, описывающих лингвистические особенности текста. Данные метрики направлены на изучение стиля предполагаемого слуха.

2.2. Вовлеченность пользователя

2.2.1. Оригинальность

$$\text{Оригинальность} = \frac{\text{Количество твитов}}{\text{Количество ретвитов}} \qquad (26)$$

Оригинальность — это показатель того, насколько оригинальны сообщения пользователя в Twitter. Как показано в уравнении выше, оригинальность рассчитывается соотношением количества исходных твитов, которые произвел пользователь, и количеством раз, когда пользователь просто переписывал чужой оригинальный твит. Чем больше это соотношение, тем более изобретательным и оригинальным является пользователь. И наоборот, более низкое отношение указывает на противоположное поведение пользователя [24].

Алгоритм на псевдокоде:

1. **Procedure** Originality (twt_count, retwt_count):
2. Оригинальность - процент оригинальных твитов, а не ретвитов;
3. **Вход**: количество твитов пользователя и количество ретвитов пользователя;
4. **Выход**: вещественное число, определяющее долю оригинальных твитов пользователя относительно суммы оригинальных твитов и ретвитов;
5. **ЕСЛИ** количество_ретвитов = 0:
6. **Возвращаем значение True**
7. **ИНАЧЕ**:
8. **RETURN** (количество твитов) / (количество ретвитов)

2.2.2. Правдоподобие

$$\text{Правдоподобие} = \begin{cases} 1 - \text{Аккаунт подтвержен} \\ 0 - \text{Аккаунт не подтвержден} \end{cases} \qquad (27)$$

Каждый пользователь в сети Twitter маркируется самой компанией Twitter для обозначения официальных страниц пользователя. Будь это страничка какой-то музыкальной группы или страница популярного СМИ.

Соответственно если пользователь имеет данную маркировку, то правдоподобность этого пользователя 1, иначе 0.

Алгоритм на псевдокоде:

1. **Procedure** Credibility (verified:bool):
2. Credibility: Достоверность - определяет наличие отметки Verify для пользователя;
3. **Вход**: булева переменная, определяющая доверие к пользователю от системы Twitter;
4. **Выход**: 1, в случае если функция равно TRUE, иначе 0;
5. **ЕСЛИ** verified ==True:
6. **Возвращаем значение True**
7. **ИНАЧЕ:**
8. **Возвращаем значение False**

2.2.3. Влияние

$$\text{Влияние} = \#\text{Количество подписчиков} \tag{28}$$

Влияние измеряется простым количеством людей, читающих пользователя. Предположительно, чем больше пользователей, тем более влиятельным он или она является

1. Influence: Поток - Количество подписчиков пользователя;
2. Вычисление не требуется.

2.2.4. Роль

$$\text{Роль} = \frac{\text{Количество подсписчиков}}{\text{Количество подписок}} \tag{29}$$

Это количество человек, которые читают пользователя деленное на количество человек, которые пользователь читает. Из этого вытекают 2 роли:

1) Распространитель. Если количество подписчиков больше подписок, то есть коэффициент роли больше 1;
2) Получатель, иначе.

Алгоритм на псевдокоде:

1) **Procedure** Role(followers,followees):
2) Role: Роль - определяет роль пользователя в зависимости от соотношения подписчиков и подписок;
3) **Вход**: количество подписчиков и количество подписок пользователя;
4) **Выход**: вещественное число определяющее соотношение подписок и подписчиков пользователя;
5) **RETURN** (Followers/Followees)

2.2.5. Вовлеченность

$$\frac{\text{Подсписчики} + \text{Подписки} + \text{Ответы} + \text{Меток "Нравится"}}{\text{Годы существования аккаунта}} \quad (30)$$

Это то на сколько пользователь был активен за все время с даты регистрации (в годах). Если меньше 1, то используется (0.5).

В последствии данные идентификаторы вместе с лингвистическим разбором и динамикой распространения будут направлены в качестве входных данных для нейронной сети. На выходе мы получаем вероятность достоверности слуха.

Алгоритм на псевдокоде:

1. **Procedure** Engagement (tweets:int, retweets:int, replies:int, favorites:int, acc_age:int):
2. Engagement: Вовлеченность - определяет активность пользователя за время существования аккаунта;

3. **Вход**: количество твитов, количество ретвитов, количество ответов, количество сообщений в избранном и количество целых лет существования аккаунта.
4. **Выход**: вещественное число значение которого определяет степень вовлеченности пользователя в систему Twitter;
5. **RETURN**
   (Количество_твитов+Количество_ретвитов+Количество_ответов+Количество_ «избранных»)/(Количество_лет)

2.3. Динамика распространения.

Особенности распространения фиксируют временную диффузионную динамику слухов. Было обнаружено, что в общей сложности 6 особенностей распространения существенно влияют на результаты моделей. В порядке убывания вклада эти функции: доля диффузии с низким уровнем к высокому, доля узлов в наибольшем компоненте связности (LCC), отношение средней глубины к ширине, отношение новых пользователей, отношение оригинальных твитов, доля твитов, содержащих внешние ссылки, и доля изолированных узлов. Все эти функции получены из графа диффузии слухов. Прежде чем мы подробно опишем эти особенности, нам нужно объяснить, как был создан граф диффузии [24].

Предполагаемая по времени диффузия довольно простой способ отхватить диффузию твитов - анализ пути ретвита этих твитов. Поскольку каждый твит и ретвит помечены меткой времени, можно отслеживать временную диффузию сообщений в Twitter. Однако API Twitter не предоставляет истинный путь ретвита твита. На Рисунок 6 показано дерево ретвитов, предоставляемое API Twitter. Можно увидеть, что все ретвиты указывают на оригинальный твит. Это не отражает истинного дерева ретвитов, поскольку во многих случаях пользователь ретвитит ретвиты других пользователей, а не оригинальный твит. Но, как вы можете видеть на рис. 3,

все заслуги отдаются пользователю, который твитнул оригинальный твит, независимо от того, кто ретвитнул кого.

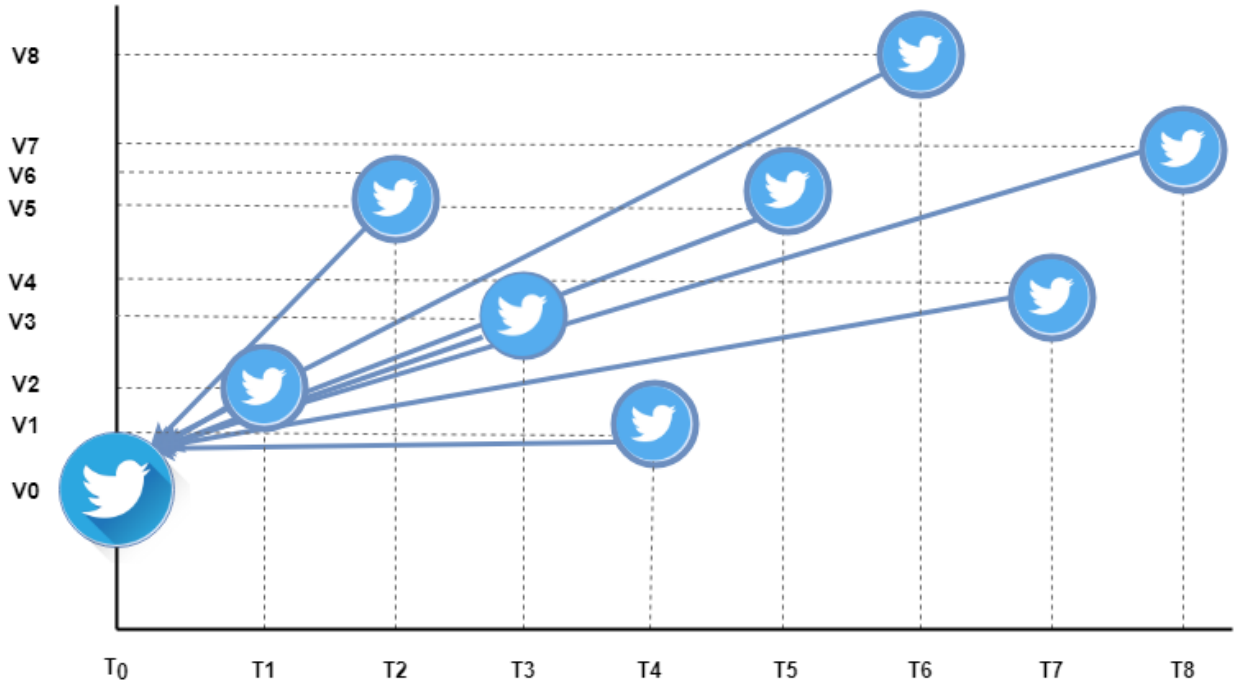

Рисунок 3. Дерево ретвитов предоставляемое системой Twitter

К счастью, мы можем вывести истинный путь ретвита твита, используя график последователя Twitter. На рис. 4 показано, как это достигается. Верхняя панель на рисунке показывает путь ретвита, предоставляемый API Twitter. Средняя панель показывает, что Нижний пользователь является последователем среднего пользователя, но не Верхнего пользователя (пользователя, который написал оригинальный твит). Наконец, третья панель показывает, что, используя эту информацию и тот факт, что Нижний пользователь ретвитнул после среднего пользователя, можно сделать вывод, что Нижний человек ретвитнул, должно быть, ретвитнул среднего человека, а не Верхнего человека.

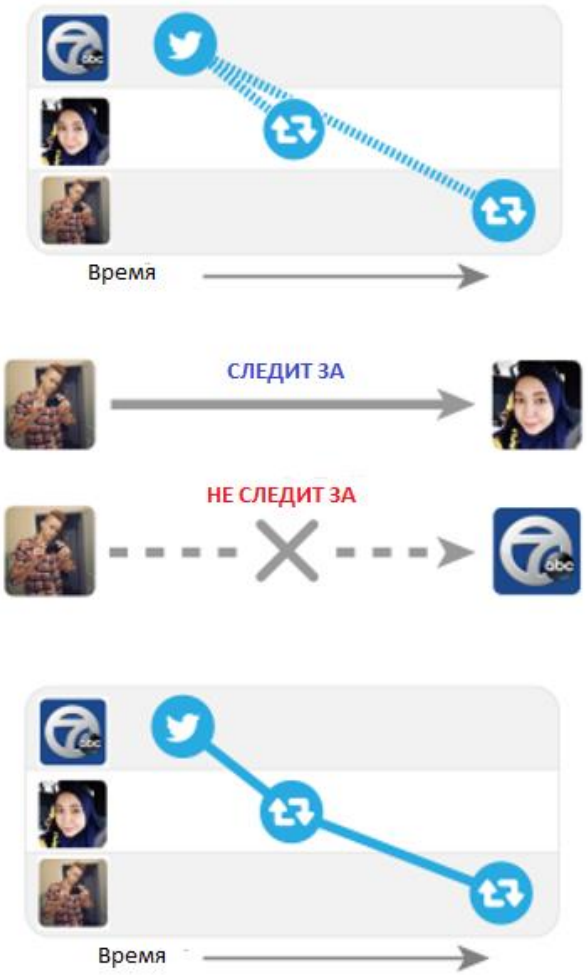

Рисунок 4. Метод восстановления истинного графа ретвитов

Этот метод восстановления истинного графа ретвита называется диффузией, основанной на времени.

Используя этот метод, мы можем преобразовать наше гипотетическое дерево ретвитов показанного на рис. 3 для более точного представления об истинном дереве ретвита, показанного на рис. 5. Обратите внимание, что слух состоит из многих деревьев ретвитов [24].

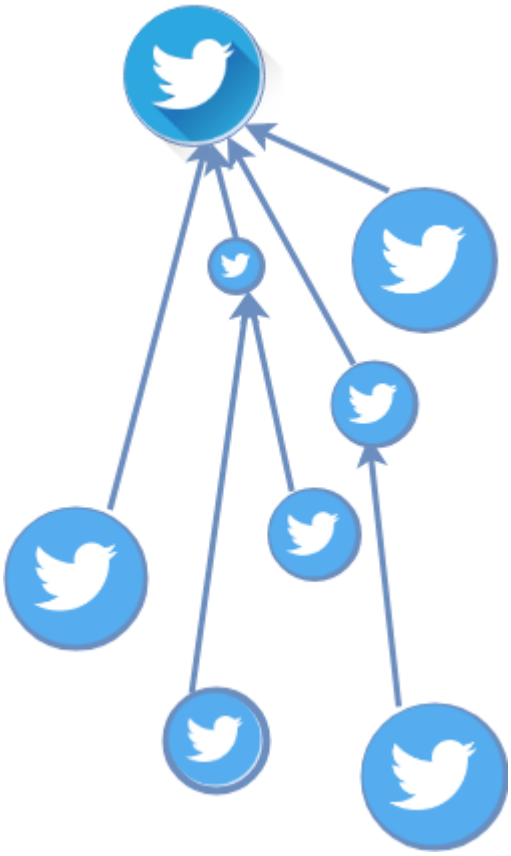

Рисунок 5. Исправленная версия пути твита

Заметим, что диффузия состоит из нескольких диффузионных деревьев, выведенных по времени. Используя эти предполагаемые временем диффузионные деревья, мы можем проследить распространение слухов через Twitter и извлечь информативные особенности о природе распространения слухов. Далее подробно объясним эти особенности.

Для определения динамики распространения необходимо построить дерево диффузий и провести его анализ.

1. **Procedure** tree(list_of_tweets):
2. Построение дерева диффузионных событий
3. **Вход:** Список твитов содержащих как минимум ID твита и текст, чтобы получить список ретвитов.
4. **Выход:** Дерево диффузионных событий и его свойства для последующего определения параметров.
5. **FOR** каждый твит в списке твитов:

```
With_url = 0;
LHD = [];
ListOfRetweted:= api.GetRetweets(ID_твита);
Isolated:=ListOfRetweted;
#Лист ретвитов отсортирован по времени;
IF ListOfRetweted не пустой массив:
 Ширина_Дерева := len(ListOfRetweted);
 FOR твит в списке ретвитов:
#Проверка читает ли ретвитнувший пользователь того, кого ретвитнул;
  IF твит в Список_Подписок:
   RETURN True;
  ELSE:
#Поиск пользователя в подписках от которого пользователь сделал ретвит;
  ListOfFriendWithRetweets: = (ListOfRetweted) & (Список_подписок)
   IF ListOfFriendWithRetweets содержит одну запись:
     Добавить узел к этой записи;
     Определить присутствие эвента от низкого к высокому;
   ELIF ListOfFriendWithRetweets содержит более одной записи:
     Добавить узел к узлу, стоящему по времени перед этим в списке ListOfRetweted;
     Определить присутствие эвента от низкого к высокому;
  IF твит содержит ссылку на внешний источник:
   With_url:= with_url+1;
  IF твит не ретвитнут и на него не получен ответ, то:
   Isolated := Isolated – Id_твита
Isolated_count := len(Isolated)
```

Чтобы описать дерево событий приводятся 6 функций. Далее идет вызов функций, описание которых приводится ниже.

2.3.1. Доля диффузии от низкого до высокого уровня

Каждому ребру в графе распространение слухов (как показано на рис. 8) соответствует событие диффузии. Каждое событие диффузии происходит между двумя узлами. Диффузионные события являются направленными (в направлении времени), с информацией, распространяющейся от одного узла к другому в течение долгого времени. Мы будем называть узел, который выталкивает информацию (т. е. влияет), отправитель и узел, который получает информацию (т. е. тот, на кого влияют), получатель.

Доля диффузионного признака от низкого до высокого измеряет долю диффузионных событий, когда диффузия была от отправителя с меньшим влиянием до приемника с более высоким влиянием (см. уравнение ниже). Влияние пользователя соответствует количеству его подписчиков. Чтобы проиллюстрировать это далее, рис. 6 показывает расширенную версию дерева диффузии, показанного на рис. 5, где размер узлов соответствует влиянию пользователей. Здесь, мы можем более ясно увидеть концепцию низко-высокой диффузии. Например, диффузия между вторым и третьим узлами (слева).

$$\text{Низкая} - \text{Высокая Диффузия} = \frac{\#\ \text{Низкие}-\text{высокие диффузии}}{\#\ \text{Все диффузионные события}} \qquad (31)$$

Рисунок 6. Расширенная версия дерева диффузии

В качестве примера низкой-высокой диффузии рассмотрим следующую ситуацию:

Допустим у нас имеет пользователь А, у которого 10 тыс. подписчиков и пользователь Б, у которого 10 подписчиков. В данном случае низкая-высокая диффузия произойдет только в том случае, если пользователь А сделает ретвит твита пользователя Б, но не наоборот.

Эта функция очень информативна, поскольку она захватывает феномен очевидца, который распространен во время реальных событий. В нем также подчеркивается роль Twitter в качестве источника экстренных новостей через очевидцев на местах. Как мы объясним позже в этой главе, эта функция является наиболее предсказуемым из правдивости слухов по сравнению с любой другой особенностью сама по себе.

1. **Procedure** LHD(Количество Низких-Высоких диффузий, Все диффузии):
2. Определение эвента между двумя событиями
    a. Низкая-Высокая Диффузия = (Низкие-высокие диффузии) / (Все диффузионные события);
3. **Вход:** количество событий, в котором произошла низкая-высокая диффузия и общее количество диффузионных событий(ретвитов)
4. **Выход:** вещественное число, обозначающее долю НВД в общем количестве диффузионных событий;
5. **RETURN** Количество Низких-Высоких диффузий / Все диффузии

2.3.2. Доля узлов в наибольшей компоненте связности.

Связный компонент графа — это подграф, в котором узел доступен из любого другого узла. Самый большой компонент связности (LCC) это компонент с наибольшим числом узлов. В графике распространения Twitter LCC соответствует оригинальному твиту с наибольшим количеством ретвитов.

Доля узлов в характеристике LLC измеряет соотношение в узлах LLC слухов график диффузии, за общее количество узлов в графе диффузии. Эта функция захватывает самую длинную цепочку разговоров слухов в Twitter.

1. **Procedure** FNLCC(Количество узлов в LCC, Все узлы):
2. В графике распространения twitter LCC (longest connected component) соответствует оригинальному твиту с наибольшим количеством ретвитов;
3. **Вход:** количество узлов в наибольшем компоненте связности и количество узлов в графе диффузионных событий;
4. **Выход:** вещественное число, обозначающее долю узлов в НКС в общем количестве узлов в графе диффузионных событий;
5. **RETURN** Количество узлов в LCC / Все узлы

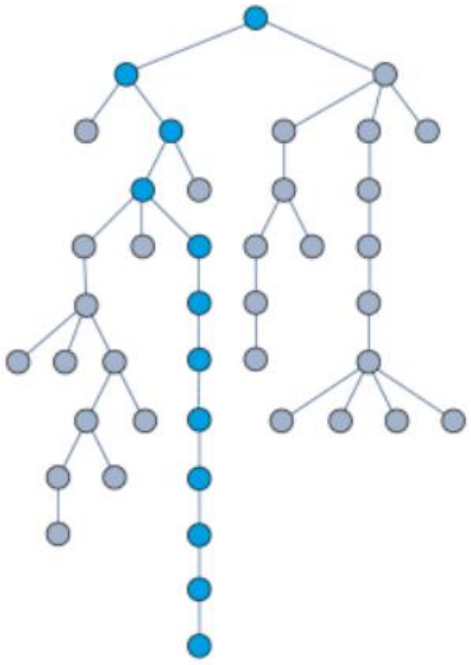

Рисунок 6. Пример LLC в графе диффузии

Доля узлов в наибольшем компонентном компоненте связности измеряет отношение узлов в наибольшем Связном компоненте графа диффузии слухов к общему числу узлов в графе диффузии и принимает значение от 0 до 1. [24]

$$\text{Доля узлов в } LCC = \frac{\text{Количество узлов в } LCC}{\text{Все узлы}} \quad (32)$$

Эта функция захватывает самую длинную цепочку разговоров слухов в Twitter и указывает на то на сколько глубоко распространился слух.

2.3.3. Среднее отношение глубины к ширине.

Форма диффузионного графа может многое рассказать о природе диффузии. Особенность, среднее отношение глубины к ширине — это попытка количественно оценить форму диффузионного графика слухов. Глубина дерева диффузии определяется как самый длинный путь от корня до листа. Ширина дерева определяется как общее число содержащихся в нем узлов. Поскольку каждый граф диффузии слухов состоит из многих диффузионных деревьев, мы усредняем отношение глубины к ширине диффузионных деревьев в слухе. Здесь N относится к числу диффузионных деревьев в слухе.

$$\text{Среднее отношение глубины к ширине} = \frac{\sum^{N}(\frac{\text{Колчество ячеек в } LCC}{\text{Все ячейки}})}{N} \quad (33)$$

1. **Procedure** ADBR(Количество узлов в LCC, Все узлы, количество деревьев):
2. Глубина дерева диффузии определяется как самый длинный путь от корня до листа. Ширина дерева определяется как общее число содержащихся в нем узлов.
3. **Вход:** количество узлов в самом длинном пути от корня до листа, общее количество узлов в дереве и количество деревьев для одного слуха;
4. **Выход:** вещественное число обозначающее среднее отношение глубины к ширине;
5. **RETURN** Сумма(Количество узлов в LCC / Все узлы)/ количество деревьев

2.3.4. Соотношение оригинальных твитов

Простая мера того, насколько увлекательным, интересным и оригинальным является разговор о слухах. Это измеряется соотношением

новых твитов и ответов (т. е. не ретвитов) в графе распространения слухов. Уравнение ниже показывает точную формулу, используемую для вычисления этой функции.

$$\text{Оригинальные твиты} = \frac{\text{Твиты+Ответы}}{\text{Твиты+Ответы+Ретвиты}} \quad (34)$$

1. Procedure original(Твиты, Ответы, Ретвиты):

2. Простая мера того, насколько увлекательным, интересным и оригинальным является разговор о слухах. Это измеряется соотношением новых твитов и ответов (т. е. не ретвитов) в графе распространения слухов.

Original Tweets = (Твиты + Ответы) / (Твиты + Ответы + Ретвиты)

3. Вход: Количество твитов, ответов и ретвитов в слухе;

4. Выход: вещественное число обозначающее среднее отношение глубины к ширине;

5. RETURN (Твиты + Ответы) / (Твиты + Ответы + Ретвиты)

2.3.5. Доля твитов, содержащих внешние ссылки

Твиты могут содержать ссылки на источники за пределами Twitter. Очень часто для твитов, которые говорят о реальных ситуациях и событий, ссылки на новостные организации или других социальных медиа. При изучении слухов в Twitter интуитивно понятно, содержат ли твиты, которые делают утверждения, ссылки на другие источники (т. е. есть ли другие источники, подтверждающие утверждения). Хотя в настоящее время мы не отслеживаем распространение слухов за пределами Twitter, благодаря этой функции мы можем иметь очень грубое приближение фактора подтверждения.

Уравнение ниже показывает, как рассчитывается эта функция.

$$\text{Доля твитов, содержащих внешние ссылки} = \frac{(\text{твиты с ссылками})}{\text{все твиты}} \quad (35)$$

1. **Procedure** with_url(Количество узлов с ссылками, Все узлы):

2. Доля твитов, содержащих внешние ссылки = узлы с ссылками / Все узлы
3. **Вход:** количество узлов, содержащих ссылки на внешние ресурсы (т.е. не в сети Twitter), количество всех узлов
4. **Выход:** вещественное число, обозначающее долю узлов, содержащих внешние ссылки;
5. **RETURN** Количество узлов с ссылками / Все узлы

2.3.6. Доля изолированных узлов.

Не все твиты будут ретвитнуты или получат ответ. Согласно исследованию аналитической компании Sysomos в социальных сетях, около 71% твитов никогда не получают ответа. Твиты, которые не получают ответа, могут быть индикаторами пользователя, который не имеет значения, сообщение, которое неинтересно, или чрезмерное насыщение подобных сообщений в Twitter. Все эти факторы показывают что-то о природе твита, и при анализе для всех твитов, содержащихся в слухах, он может выявить что-то о природе слуха.

$$\text{Изолированные узлы} = \frac{\text{Узлы без ответа и ретвита}}{\text{все узлы}} \tag{36}$$

Твит, который не получает ответов или ретвитов, отображается как изолированный узел в графике распространения слухов. Эта функция захватывает долю всех узлов в графе диффузии слухов, которые изолированы.

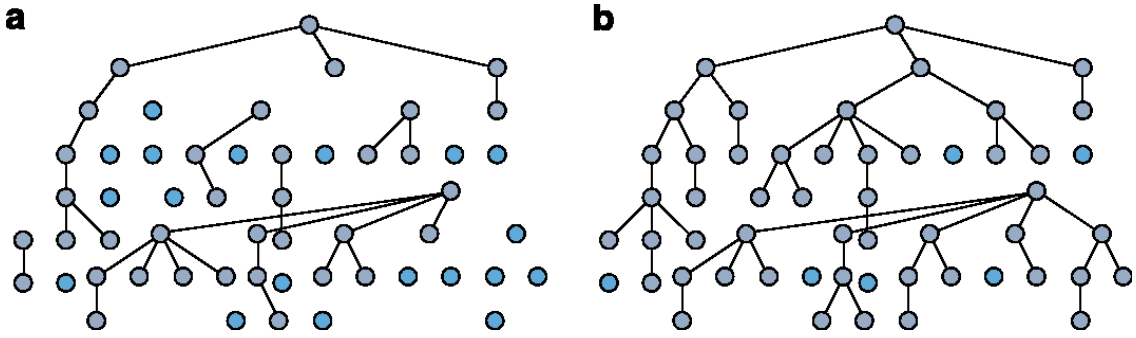

Рисунок 8. Примеры графов с изолированными ячейками

1. **Procedure** isolate(Количество изолированных узлов, Все узлы):

2. Твит, который не получает ответов или ретвитов, отображается как изолированный узел в графике распространения слухов, т.е. можно считать, что требуется проверить листья дерева на наличие ответов.
3. **Вход:** количество изолированных узлов в дереве диффузий и общее количество узлов;
4. **Выход:** вещественное число, обозначающее долю изолированных узлов в дереве диффузий;
5. **RETURN** Количество изолированных узлов / Все узлы

Динамика распространения позволяет получить оценку распространения слуха относительно временного промежутка. Оценка данных параметров отображает

2.4. Алгоритм использования модели нейронной сети

Основой для создания хорошей нейронной сети является правильный подбор параметров для получения лучших показателей нейронной сети. Существуют методы для получения данных параметров и оценки нейронной сети. В зависимости от направления работы нейронной сети выполняется набор действий, выявляющий корректные параметры, и вычисляет оценку работы. Опишем данные действия:

2.4.1. Определение параметров Модели нейронной сети

Определение параметров модели нейронной сети позволяет создать максимально подходящую для задачи модель нейронной сети, которая позволит получить лучшие результаты;

Вход: очищенный набор данных, разделенный на тестовую и обучающую выборки;

Выход: определенный набор параметров модели нейронной сети позволяющий получить лучшие результаты на обучающей выборке;

1. Выполнить поиск по сетке с предустановленными параметрами с собранными данными;
2. Выполнить оценку найденных параметров с помощью визуализации;

3. Выполнить оценку точности работы рассматриваемых методов метрикой accuracy;

$$accuracy = \frac{P}{N}, \quad (37)$$

где P – правильно классифицированные, а N – размер всей выборки.

4. Выполнить оценку точности работы рассматриваемых методов с помощью метрики MSE;

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2 \quad (38)$$

5. Выполнить оценку точности работы рассматриваемых методов с помощью метрики MAE;

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y(i) - \hat{Y}(i)\right| \quad (39)$$

6. Выполнить оценку точности работы рассматриваемых методов с помощью метрики R^2;
7. Выполнить анализ параметров модели и подобрать оптимальные параметры;

2.4.2. Обучение Модели нейронной сети

Обучение модели нейронной сети позволяет создать на имеющихся данных систему по определению слуха.

Вход: набор твитов удовлетворяющих условию поиска. Заранее известно является ли твит слухом или нет.

Выход: модель нейронной сети для поставленной задачи определения наличия слуха в твите.

1. Выполнить обучение полученной модели;
2. Показать точность классификации с помощью матрицы ошибок;
3. Использовать метрику Precision (Точность) по формуле 18;

4. Использовать метрику Recall (Полнота) по формуле 19;
5. Использовать F-меру по формуле 22;
6. Провести тестирование на тестовых данных;
7. Реализовать сохранение модели для последующих использований пропуская процесс обучения;
8. Оптимизировать модель;
9. Сохранить модель.

Использование подобной конструкции позволяет получить многофункциональную модель нейронной сети, которая не требует переобучения, а также предоставляет все необходимые оценки работы модели.

ГЛАВА 3. ИСЛЕДОВАНИЕ ПРЕДМЕТРОЙ ОБЛАСТИ И ФУНКЦИОНАЛ СИСТЕМЫ

3.1. Структура и состав программного обеспечения

При реализации приложения необходимо решить вопросы реализации, такие как функции, пользовательский интерфейс, база данных и доступ к ней.

Приложение для информационной системы является толстым клиентом с частично скомпилированным тонким.

Рисунок 10. Архитектура развертывания

В состав программного обеспечения входит: База данных, приложение с графическим интерфейсом, нейронная сеть.

Приложение выполняет данные функции:

a. Обработка пользовательского запроса, введенного в графическом интерфейсе;
b. Выполнение запроса к сети Twitter для сбора данных;
c. Хранение данных в базе данных;
d. Очистка и предобработка данных для работы с нейронной сетью;
e. Пропуск данных через нейронную сеть и получение вероятности наличия слуха в предоставленных твитах;
f. Вывод результатов в графический интерфейс.

3.2. Средства разработки программного обеспечения

Рассмотрим список инструментов, которые необходимы для построения приложения для определения идентификаторов пользователя.

1) Python – язык программирования.
2) PyCharm IDE – Среда программирования с открытым кодом;
3) Twitter API;
4) Библиотека Keras для создания нейронной сети;
5) Библиотеки для создания пользовательского интерфейса;
6) Библиотеки для Python (Matplotlip, scikit-learn, numpy…);
7) Инструменты быстрого анализа и визуализации данных (Deductor, matlab, wolfram…).

Данный набор инструментов поможет в решении многих не тривиальных задач, начиная от хранения данных и их визуализации, до полноценной организации приложения.

С данной части этой работы описаны причины почему было решено взять именно эти библиотеки. Стоит отметить, что большинство библиотек работает в основе на других менее популярных (не всегда) библиотеках.

Virtualenv – Библиотека для работы с файловой системой. Необходим для создания виртуальных окружений в python, внутри которых может, например, использоваться другая версия python (не та, что установлена в системе как основная), свой, особый, набор модулей и приложений (нужных под конкретную задачу).

Python-twitter – библиотека для работы с API Twitter. Предоставляет полный доступ к API и к открытым данным пользователей, включая тексты сообщений и мета данные пользователя.

Scikit-learn – это библиотека для машинного обучения на языке программирования Python с открытым исходным кодом. С помощью нее можно реализовать различные алгоритмы классификации, регрессии и

кластеризации, в том числе алгоритмы SVM, случайного леса, k-ближайших соседей и DBSCAN, которые построены на взаимодействии библиотек NumPy и SciPy с Python.

Matplotlib – Библиотека для визуализации данных. Поддерживает самые современные методы визуализации. Так же поддерживает все популярные структуры хранения данных.

Keras – Высокоуровневая библиотека для работы с нейронными сетями написанная на Python. Она работает поверх Theano и TensorFlow.

a. Позволяет быстро экспериментировать с параметрами (через общую модульности, минимализм и расширяемость);
b. Поддерживает как сверхточных сетей и рекуррентных сетей, а также комбинации этих двух;
c. Поддерживает произвольные схемы подключения (в том числе множеством входов и множеством выходов обучения);
d. Работает без проблем на CPU и GPU.

Pandas - программная библиотека на языке Python для обработки и анализа данных. Работа pandas с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня. Pandas предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами.

Anytree - программная библиотека на языке Python для структуризации данных в виде деревьев.

3.3. Сбор данных

Как было описано в главе 1.3, за сбор данных отвечает взаимодействие приложения MOZDEH и скрипта, написанного на python, для взаимодействия с API сети Twitter. Процесс получения данных представлен на изображении ниже:

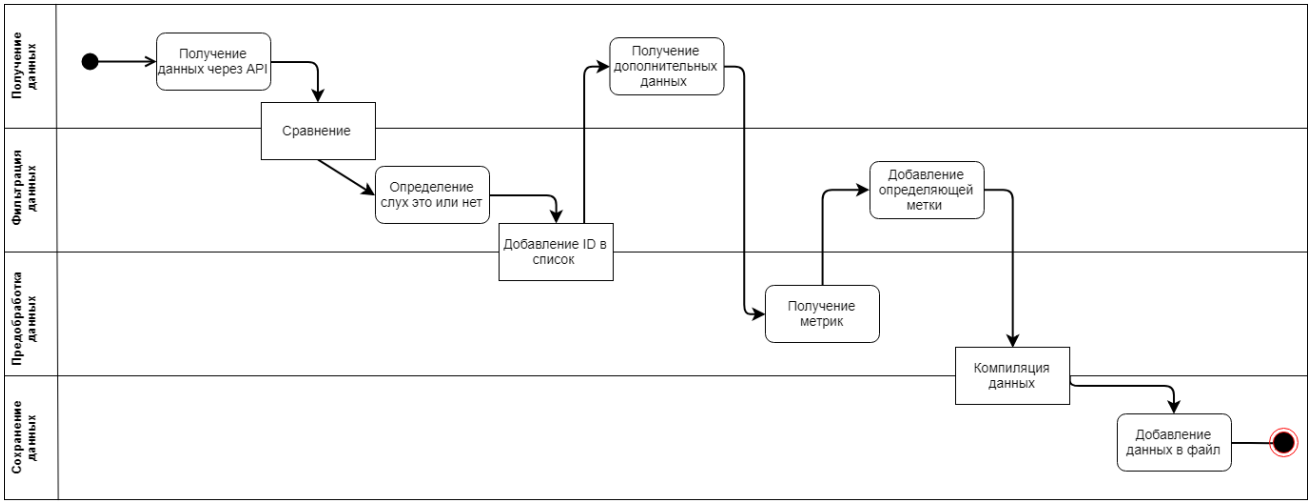

Рисунок *11.* Схема получения данных для обучения модели

Рассмотрим процесс получения данных:

1) Подключение к API. Для подключения к API Twitter необходимо выполнить регистрацию в dev версии Twitter и зарегистрировать приложения. После регистрации в личном кабинете будут доступны ключи для подключения к API. Подключение же осуществляется через библиотеку TwitterPython или Mozdeh;

2) Запрос. Зная тематику новости, которая потенциально является слухом, необходимо выполнить запрос к сети Twitter. В качестве ответа придет выборка твитов;

3) Определение слух это или нет. Для определения наличия слуха применялось сравнение с новостными агрегаторами;

4) Для работы приложения необходимы дополнительные данные, часть которых не поступает вместе с запросом. Дополнительные данные получаем через дополнительный скрипт;

5) Вручную проставлялась метка 1 если слух и 0 если не слух;

6) Данные для работы модели сохраняются в файл CSV для удобного доступа к файлам.

В конечном итоге было собрано более 3000 экземпляров данных, среди них 1196 это слухи, а остальные ни слухи более подробно в таблице ниже.

Таблица 3. Набор данных по слухам в обучающей выборке

Тематика слухов:	Количество		
Париж	Оригинальные:	59	121
	Ретвиты:	62	
Школа	Оригинальные:	128	207
	Ретвиты:	79	
Выборы	Оригинальные:	291	376
	Ретвиты:	85	
Остальное	Оригинальные:	175	492
	Ретвиты:	317	
	Всего:		1196

Остальные 1830 записей это комбинация обычных сообщений, прошедших проверку на наличие слуха, но не являющиеся слухом.

3.4. Структура хранения данных

Рассмотрим структуру данных необходимых для получения через API Twitter и зачем они нужны.

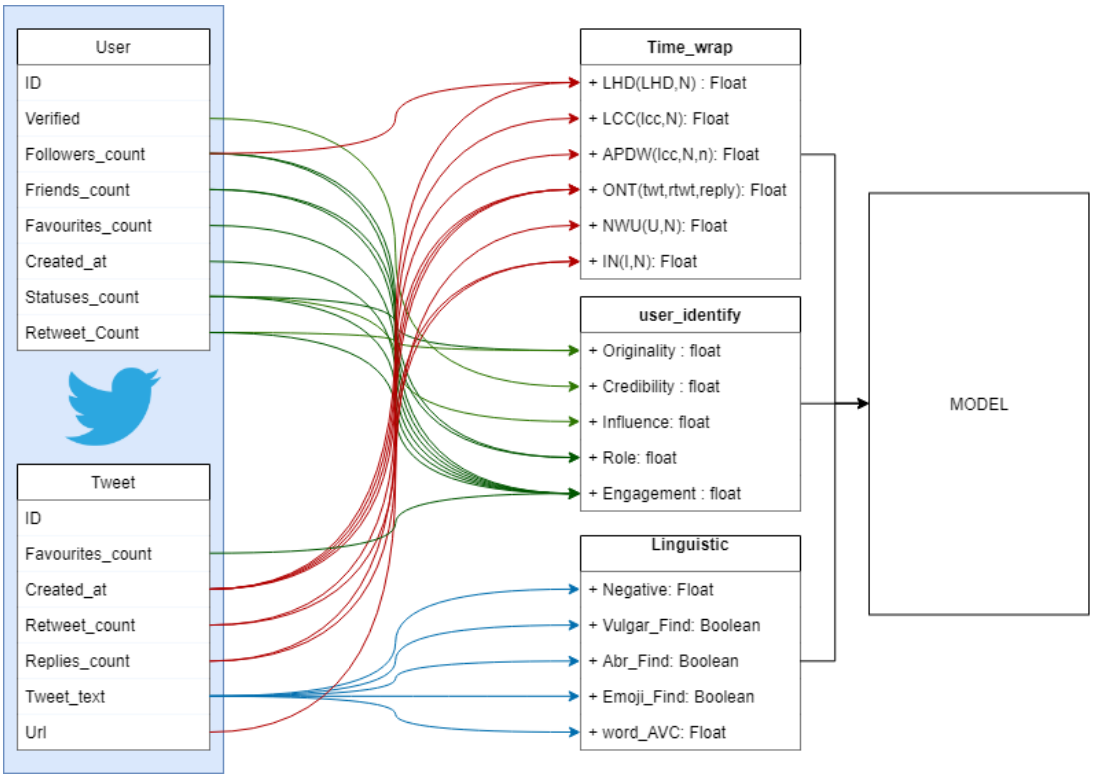

Рисунок 12. Схема заимствования данных из API для получения метрик

На рисунке выше изображена структура данных, которые необходимы при получении данных из сети Twitter.

Данные поступают в формате JSON из которого при помощи написанных функций выбираются необходимые данные и записываются в базу данных. Данные используются для вычисления метрик и поэтому необходимо привести их к нужному формату.

Среди всех данных некоторые необходимо преобразовать к необходимому виду. Например, «Created_at» поступает в формате «Thu May 31 08:17:29 +0000 2018», но из этого нам необходима полная дата и время, т.е. дата 31.05.2018 и время 08:17:29. Так же необходимо разделить их на разные части, для вычисления количества лет существования аккаунта.

База данных написана в утилите SQLiteStudio, что представляет из себя приложение для просмотра, создания и изменения баз данных в формате sqlite. База данных содержит 5 таблиц, каждая содержит отдельный набор данных (Рисунок *10*).

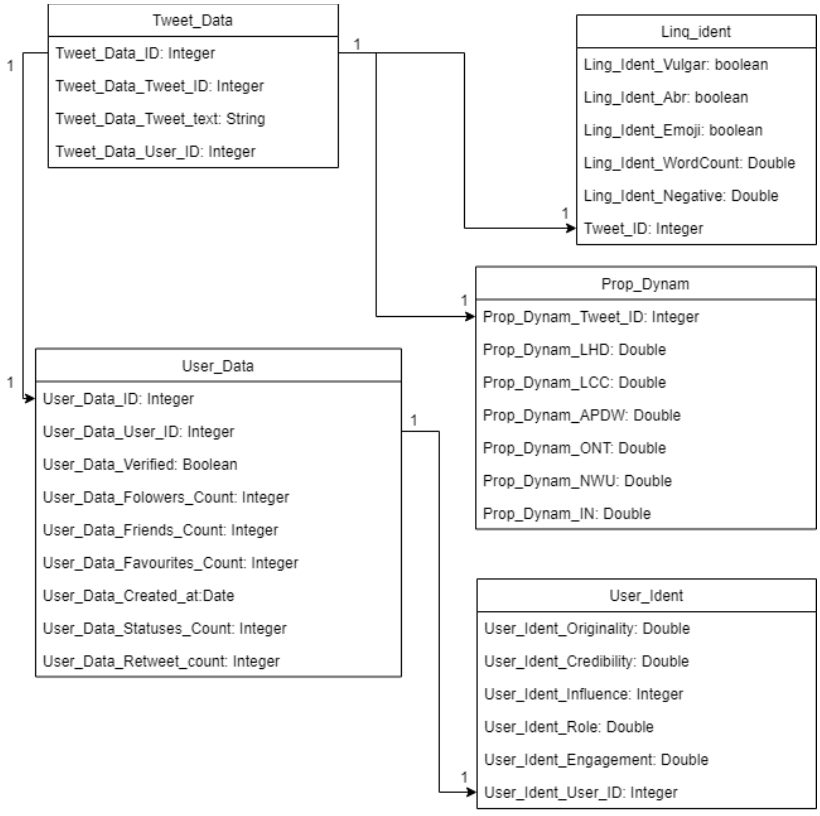

Рисунок *13.* Схема базы данных

Все данные привязаны к ID твита, т.е. в базе данных хранятся как ID твита, так и ID твита в базу данных. Подключение к базе данных осуществляется по средствам взаимодействия языка python и библиотеки SQLite3, которая позволяет осуществлять запросы в базу данных и записывать полученный результат в любой доступный в python метод хранения данных.

ГЛАВА 4. АРХИТЕКТУРА СИСТЕМЫ И ОПИСАНИЕ ЕЕ ОСНОВНЫХ КОМПОНЕНТОВ

4.1. Выбор модели нейронной сети

Для поиска параметров нейронной сети использовалась комбинация методов, которые предоставляет библиотека scikit-learn. Первый метод — это прямой поиск по сетке, а второй случайный поиск по сетке.

Для работы этих методов необходимо определить параметры для поиска. Рассмотрим варианты моделей и их параметры:

1) Multi-layer Perceptron classifier:

Параметрами многослойного персептрона выступают:

hidden_layer_sizes – количество нейронов в скрытых слоях по умолчанию 100.

Activation – функция активации. Принимает одно из четырех значений. Функциями могут выступать: identity, logistic, tanh, relu.

Alpha – параметр отвечающий за штраф от параметра L2.

batch_size – параметр, определяющий количество экземпляров в обучающей выборке на каждой эпохе.

Остальные параметры принимают значение по умолчанию.

2) Linear classifiers:

loss – Функция потерь для использования. По умолчанию 'шарнир', который дает линейный SVM. Возможные варианты: ‘hinge’, ‘log’, ‘modified_huber’, ‘squared_hinge’, ‘perceptron’.

Penalty – Штраф (он же термин регуляризации), который будет использоваться. По умолчанию используется ‘l2', который является стандартным регуляризатором для линейных моделей SVM.

alpha – Константа, умножающая член регуляризации. По умолчанию 0.0001 также используется для вычисления learning_rate, когда установлено значение "оптимальный".

Остальные параметры остаются по умолчанию.

3) Gaussian Naive Bayes: В качестве параметров принимает данные и априорные вероятности классов. Если указано, Приоры не корректируются в соответствии с данными.

4) Линейный классификатор SGD с использованием функции потерь Log. Принимает такие же параметры, что и linear classifiers.

Для каждого классификатора выполняется поиск параметров по сетке с использованием обучающих данных. Оценка классификаторов выполняется при помощи accuracy, recall, precision и f1 метрик.

Результаты f1 метрики для «лучших» моделей.

Таблица 4. Метрика f1 для полученных моделей

Logistic Regression	0.870892
Naive Bayes	0.962656
Linear SVC	0.883803
MLP	0.914046

Из данной таблицы видно, что лучше всего себя показала модель с Наивной Байевской функцией, но для окончательного решения рассмотрим график калибровки модели, который покажет какая модель ведет себя лучше.

Чтобы определить какая модель лучше необходимо изучить движение кривых на графике, чем ближе линия к центру, тем лучше откалибрована модель.

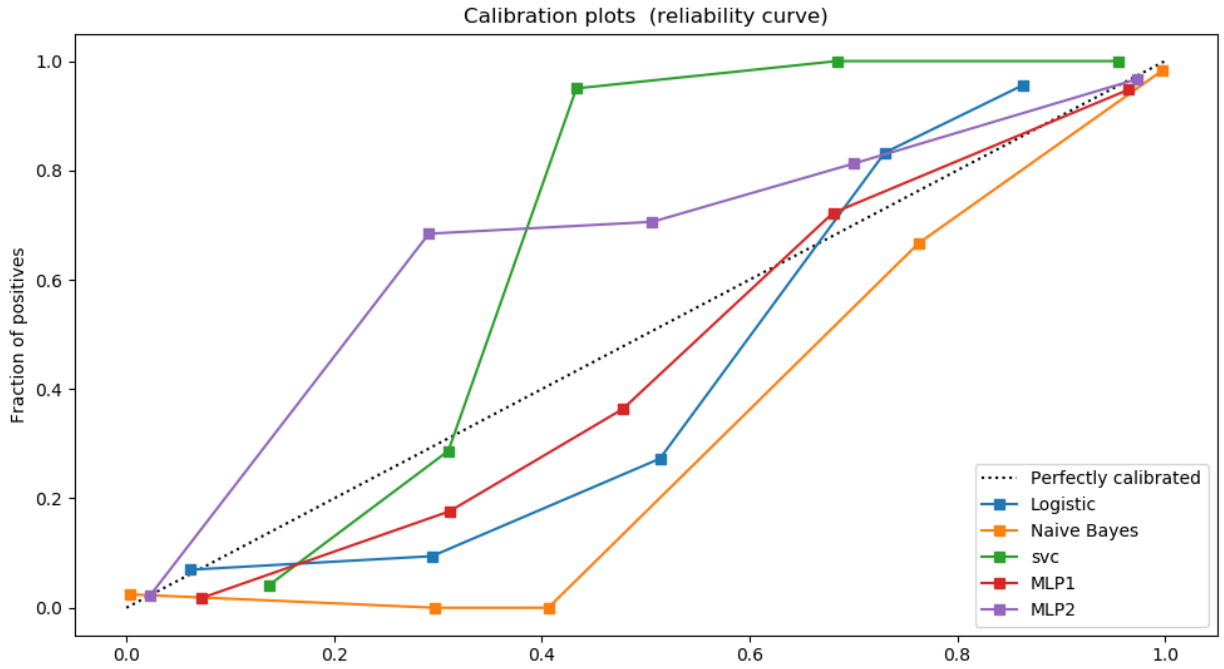

Рисунок 14. Схема базы данных

Изучив график на рис. 14 можно наблюдать, что модель с лучшим F1 показателем ведет себя плохо на графе калибровки. Так же для построения использовались два вида многослойного персептрона для двух наборов параметров, показавших себя почти одинаково. Поэтому принято решение в качестве основной модели нейронной сети взять многослойный персептрон.

4.2. Обучение нейронной сети

Обучение модели нейронной сети проходит в несколько этапов. Первым из них является загрузка и подготовка выборки для обучения. Данные содержатся в формате csv и загружаются при помощи библиотеки Pandas в формате Dataframe [19].

Следующим шагом является разбиение выборки на обучающую и тестовую выборки. Для этого необходимо определить 2 таблицы X и Y, где в X попадают все данные с параметрами, а в Y маркеры при этом сохраняя индексацию строк. При помощи специальной функции выполняется разбиение на две выборки в соотношении 70\30.

Далее задается набор параметров нейронной сети, которые были получены при помощи поиска по сетке заданных параметров. Поиск осуществляется линейно или с случайным выбором. Оценка параметров осуществляется при помощи набора метрик. Чем выше данные параметры, тем лучше работает модель, но необходимо помнить об переобучении модели.

В итоге получились следующие параметры:

{'solver': 'sgd', 'max_iter': 50, 'hidden_layer_sizes': (25,), 'alpha': 1e-6, 'learning_rate_init': 0.2, 'activation': 'tanh'}

С данными параметрами метрики получили следующие значения:

Таблица 5. Метрики оценки параметров модели

accuracy	0.927176
recall	0.931624
precision	0.897119
f1_score	0.914046

Рассмотрим результаты кросс-валидации. Кросс-валидация производилась при помощи библиотеки scikit-learn с параметрами: 5 выборок разбиения и вид кросс-валидации KFold. Итоговые результаты кросс-валидации представлены в таблице 4.

Таблица 6. Результаты Кросс-валидации

0.90322581
0.93841642
0.91788856
0.92082111
0.92058824

Используя полученные параметры проводим обучение модели при помощи данных. Далее необходимо скомпилировать модель и сохранить ее для дальнейшего использования.

4.3. Основные компоненты

Разработанная система имеет вид библиотеки Python, т.е. может использоваться как отдельно, так и предоставлять функции для других библиотек.

Приложение использует основные компоненты для разработки графического интерфейса.

Рассмотрим набор основных функций системы.

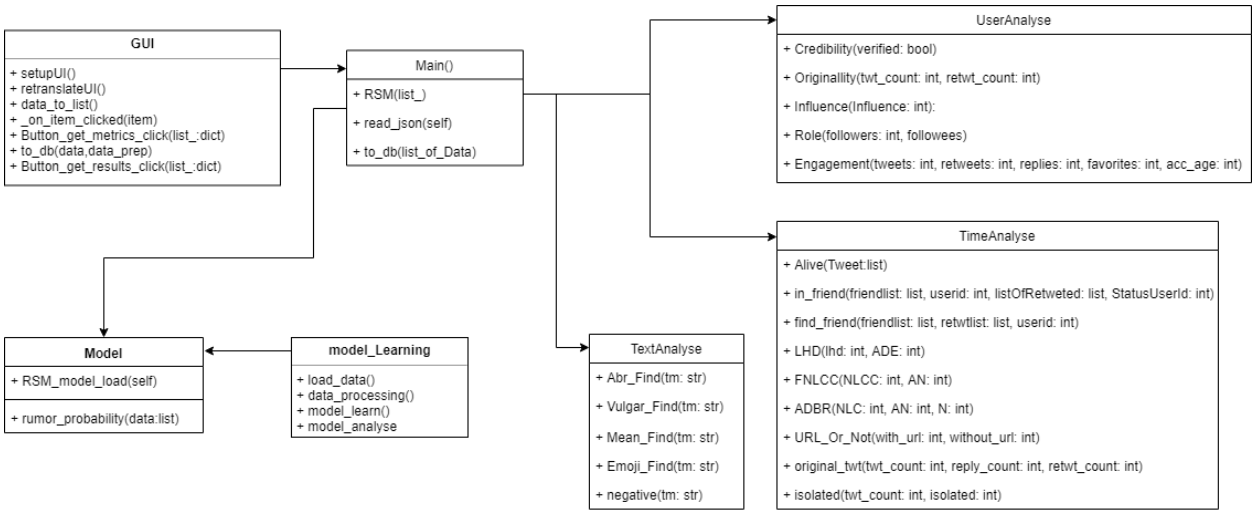

Рисунок 15. Диаграмма классов приложения

Первой основной функцией является вызов приложения и передача необходимых параметров. Основными параметрами является список параметров, которые необходимо передавать для работы в приложении, но так как получить данные параметры напрямую не всегда возможно предусмотрена функция считывания параметров из API имея всего лишь один ID. Либо считывание из подготовленного файла в формате json с специальным форматированием, который требуется заполнить вручную или вызвать функцию, которая попросит заполнить построчно данный файл.

Дальше выполняется запись в файл базы данных для постоянного доступа к данным и удобной работы.

Используя полученные данные передаем их на следующие модули:

Модуль «Анализ текста», который включает в себя функции, описанные в пункте 2.1. Среди них: подсчет слов в тексте поиск аббревиатур и сокращений слов, поиск вульгарных слов, поиск слов, указывающих на мнение, поиск смайликов.

Для каждой из функций требуется только текст твита на входе, а списки уже встроены в систему в виде текстовых файлов. Поиск выполняется при помощи регулярных выражений, что позволяет записывать в файлы части слов и ограничивать их при помощи специальных символов, таких как *,?,() и других. Данная структура позволяет охватить большой набор слов, не внося каждое отдельно. Возвращает булево значение 0 если не найдено и 1 иначе.

Модуль «Анализ пользователя» включает в себя набор функций, описанный в пункте 2.2 об оценки поведения пользователя в сети. Рассмотрим каждую функцию:

Доверие: определяется через значение параметра Verify в сети Twitter, т.е. не требует дополнительного вычисления.

Оригинальность: на входе поступают количественные данные о сообщениях пользователя за все время существования аккаунта. Данные содержат информацию о твитах и ретвитах. Оригинальность определяется как доля ретвитов в общем количестве твитов.

Поток: определяется через количество твитов.

Роль: при помощи значений количества подписок и подписчиков определяется «роль» пользователя. Если значение подписок больше, чем подписчиков, то роль определяется как «читатель», иначе «распространитель». В случае если равных параметров пользователю

применяется значение «читатель», так как велика вероятность наличия в подписчиках фейковых аккаунтов, ботов и т.д.

Вовлеченность: является оценкой всех действий пользователя за все время существования аккаунта. Для вычисления используются количественные показатели твитов, ретвитов, ответов, отметок «нравится» и дата создания аккаунта. Сумма количественных данных делится на количество лет, округленное в большую сторону.

Выполняя обработку по данным модулям модуль возвращает 5 параметров.

Модуль «Динамика распространения» отвечает за определение параметров распространения твита относительно времени, описанных в пункте 2.3.

Перед тем как вычислять параметры динамики распространения необходимо привести дерево диффузий к правильному виду. При помощи API получаем данные о ретвитах, которые включают ID, дату создания и ID пользователя, который сделал этот ретвит. По ID пользователя получаем список ID подписок и ищем среди них ID пользователя оригинального твита. Если поиск удался, то данный ID добавляется к корню дерева. Если поиск не увенчался успехом, то выполняется поиск ID пользователя оригинального твита среди подписок каждого ретвитнувшего этот твит.

Если совпадений несколько, то при помощи сортировки по времени, среди списка совпавших, выясняем кто был последним, чтобы определить к какому узлу будет добавлен данных узел.

Когда дерево построено, то можно выполнять вычисления параметров динамики распространения.

Доля диффузии от низкого до высокого уровня: для вычисления данного параметра необходимо во время построения дерева вычислять разницу в подписчиках пользователей во время добавления узлов. Просто сравнение и

итератор позволяют получить количество диффузии от низкого до высокого уровня. Зная общее количество узлов в дереве узнать долю узлов диффузии от низкого до высокого, не представляет ничего сложного.

Доля узлов в наибольшей компоненте связности. для вычисления наибольшего компонента связности необходимо отслеживать глубину дерева во время построения. Вычисление количества узлов в LCC позволяет вычислить долю в общем количестве узлов в дереве.

Среднее отношение глубины к ширине. используя полученное количество узлов в наибольшем компоненте связности и поделить на общее количество узлов в дереве, но так как один слух могут описывать несколько сообщений, то оценивается ширина относительно каждого сообщения.

Соотношение оригинальных твитов: для определения оригинальности твита необходимо выполнять проверку на наличие дополнительных комментариев к ретвиту.

Доля твитов, содержащих внешние ссылки: необходимо проводить проверку на наличие ссылок в сообщениях. Для этого с помощью API проверяем текст на наличие ссылок, которые Twitter принимает за внешние. Ссылки поступают вместе с сообщением в специальном виде сокращенной ссылки, которая генерируется автоматически, но оригинальная ссылка содержится в API.

Доля изолированных узлов. для определения изолированного узла необходимо выполнять проверку состояний твита. Всего необходимо проверять наличие: комментария, ответа, отметки «нравится» и ретвита данного сообщения. После определения количества изолированных узлов необходимо определить долю относительно всех узлов.

В дальнейшем данные, полученные в этом модуле, поступают обратно на основной модуль и происходит запись в базу данных.

В последнем модуле выполняется загрузка модели, обученной ранее при помощи подготовленных функций. Для корректной работы нейронной сети проводится дополнительное форматирование данных, полученных в предыдущих модулях. На выходе получаем значение 0 или 1 и значение вероятности, при котором было принято решение. Производится запись в базу данных, в которой можно всегда посмотреть результаты и все параметры, при которых они получились.

Данная библиотека позволяет использовать модули по отдельности, что с небольшой доработкой может помочь в ее адаптации для других социальных сетей, которые имеют подобную Twitter структуру.

4.4. Графический интерфейс

Для удобной работы с приложением была разработана графическая оболочка для приложения [18]. Написана она при помощи библиотеки PyQT5 и разработана при помощи PyQT5 Designer. Данное программное обеспечение позволяет смоделировать внешний вид приложения без необходимости писать код. Рассмотрим формы полученного приложения:

Первая и основная форма (Рисунок 12) — это главное окно программы, которое выполняет показательную функцию. На нем можно заметить, что оно делится на несколько составных частей.

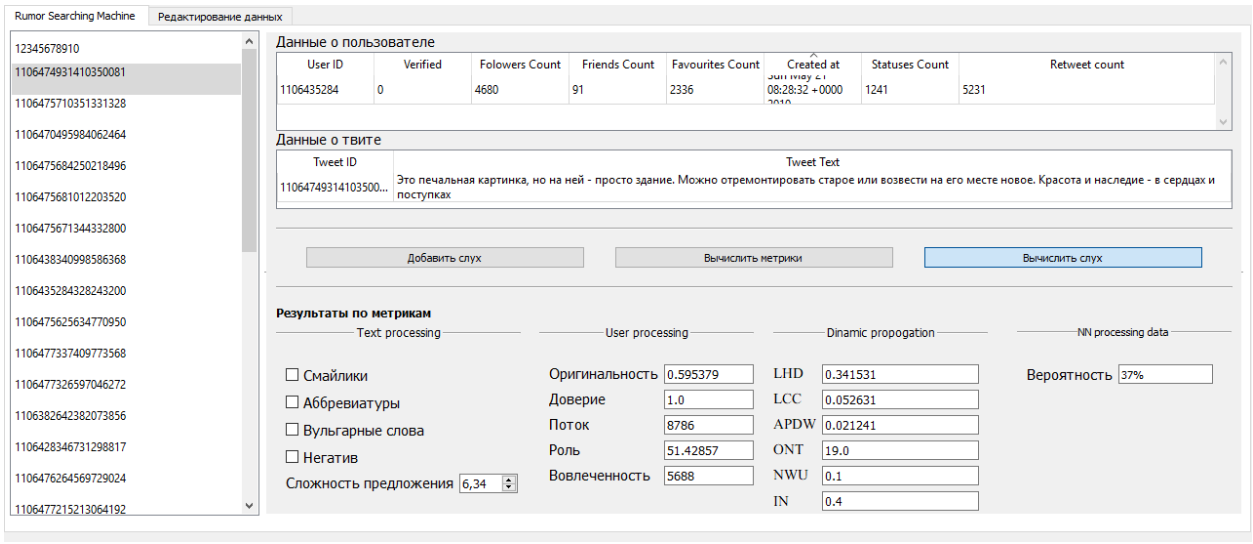

Рисунок 16. Форма «Главное окно приложения»

Первая часть — это список всех проведенных проверок твитов. Нажимая на каждую запись в списке, можно вывести все данные, которые есть на данный твит. Данные выводятся в следующую часть формы. Данная часть отвечает за загрузку данных из базы данных в таблицы. Первая таблица (2) показывает данные выгруженные из API Twitter содержащая данные о пользователе, а вторая (3) таблица показывает данные об сообщении. Под таблицами располагается контрольная панель управления с тремя кнопками. Каждая кнопка отвечает за определенное действие:

Кнопка «Добавить слух» вызывает окно «добавить», описание которого будет ниже.

Кнопка «Вычислить метрики» отвечает за обработку данных слуха в случае, если они отсутствуют. Если данные на месте, то кнопка блокируется. Вычисленные метрики поступают в базу данных и выводятся в панели ниже.

Кнопка «Вычислить слух» отвечает за работу модели нейронной сети. Полученные метрики берутся из базы данных и направляются в модель. Ответ от модели записывается в базу данных и выводится в панель NN processing data. Дополнительно выводятся метрики об оценки работы модели для данного случая.

Переключая на вторую вкладку с названием «Редактирование данных» (Рисунок 13) в которой можно добавить слова в списки, по котором осуществляется поиск. Для этого необходимо ввести слово в поле под необходимым списком и нажать кнопку Save. Слово можно добавить вместе с необходимыми символами регулярных выражений.

Удалить слово из списка можно нажатием правой кнопки мыши на необходимое слово для вызова всплывающего меню и выбрать «удалить». В этом же меню можно выбрать пункт «Переименовать» в случае ошибки.

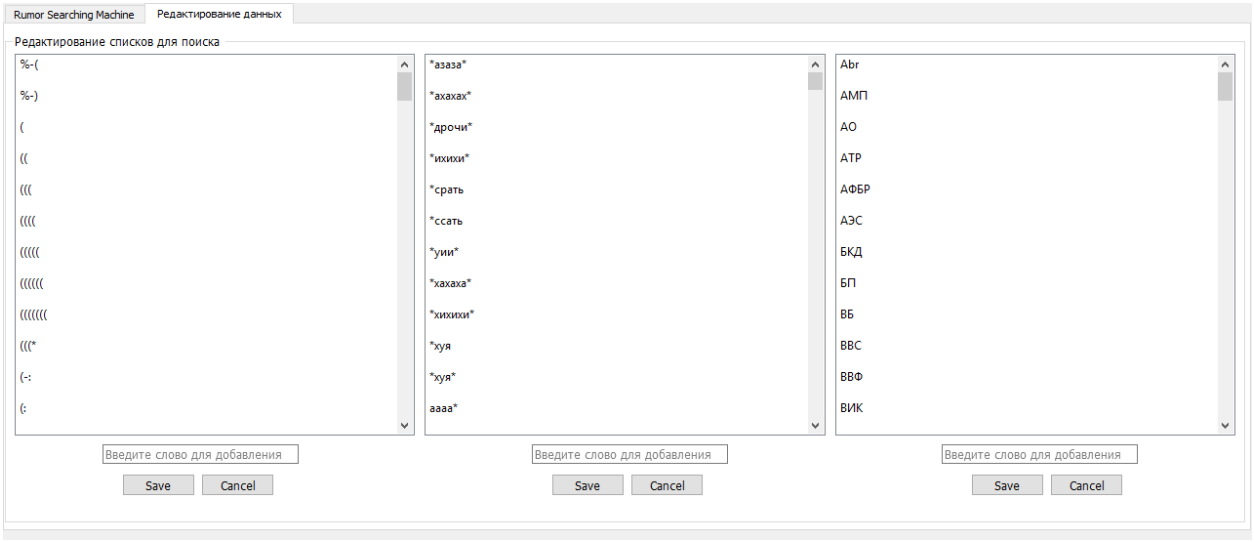

Рисунок 17. Форма «Редактирование данных»

На третьей вкладке содержатся небольшая панель настройки приложения, которые включают настройки параметров нейронной сети и др.

Форма добавления данных (Рисунок 14) выполняет функцию добавления слуха в систему. Форма состоит из двух компонентов:

Поле добавления ссылки или ID твита. Приложение автоматически получит все необходимые данные из API и внесет в базу данных. Далее новый слух появится в списке на главной форме.

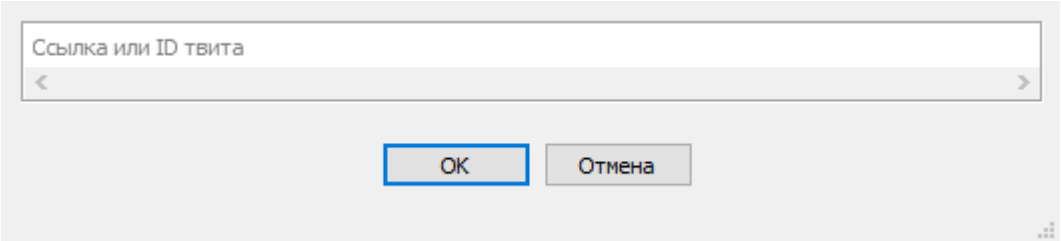

Рисунок 18. Форма «Добавления твита»

Чтобы соединить написанную библиотеку и графический интерфейс. Все дополнительные функции пишутся в скрипте python. Для этого необходимо провести конвертацию из формата ui в py. При конвертации будет создан файл со всеми настройками сделанными в Designer. Данный файл необходимо соединить с приложением и подключить к кнопкам необходимые функции.

После заполнения файла необходимыми функция необходимо произвести конвертацию py скрипта в формат exe. Для этого можно воспользоваться утилитой Auto py to exe. Утилита конвертирует файл подключая все необходимые библиотеки и помещает файл в ту же папку, что и главный скрипт графического интерфейса.

На этом приложение готово.

ВЫВОД

Достигнутые результаты в данной работе позволяют определять наличие слуха в сообщениях в социальной сети Twitter с вероятностью до 92%, что является хорошим показателем для модели. Описанное в работе приложение позволяет работать с данными как в качестве целого приложения, так и каждый модель по отдельности. Графический интерфейс предоставляет простой доступ к функциям приложения для простого пользователя.

В первой главе были рассмотрены основные методы решения подобных задач, а также литературный обзор, посвященный разбору основных научных работ в подобном направлении.

Вторая глава отражает результаты исследовательской работы, которые позволили выявить следующие метрики, которые использовались в данной работе для обучения модели нейронной сети. Описаны алгоритмы получения метрик, обучения модели и проверки качества ее работы.

В третьей главе сформулированы основные средства разработки приложения, поток данных начиная с API и заканчивая моделью нейронной сети, а также описание базы данных для хранения информации.

В последней, четвертой, главе описывается создание программного обеспечения при помощи описанных методов и средств. Представлен процесс разработки графического интерфейса.

Русский язык настолько обширный, что под одним может подразумеваться несколько значений и смыслов. Хоть модель и предоставляет высокий процент вероятности того, что ошибки при определении не произойдет, следует всегда помнить, что окончательное решение всегда стоит за человеком.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Agarwal A., Xie B., Vovsha I., Rambow O., Passonneau A.R. Sentiment analysis of Twitter data [Текст] / A.Agarwal, B. Xie, I.Vovsha, O.Rambow, A.R.Passonneau // In Proceedings of the Workshop on Languages in Social Media. Association for Computational Linguistics. - 2011. - С. 30-38.

2. Alton Y.K., Snehasish B. Linguistic Predictors of Rumor Veracity [Текст] / Y.K.Alton, B. Snehasish // Linguistic Predictors of Rumor Veracity on the. Hong Kong. - 2016. - №1. – С. 387-391.

3. Diab M., Sardar H. Rumor Detection and Classification for Twitter Data [Текст] / M. Diab, H. Sardar.- 2015. - С. 71-77.

4. Du J., Zhang Y., Luo J., Jia Y., Wei Q., Tao C., Xu H. Extracting psychiatric stressors for suicide from social media using deep learning [Текст] / J. Du, Y. Zhang, J. Luo, Y. Jia, Q. Wei, C. Tao, H. Xu // The 2nd International Workshop on Semantics-Powered Data Analytics. - Канзас сити, МО, США. - BMC Med Inform Decis Mak. - 2018;

5. Friggeri A., Adamic L.A., Eckles D., Cheng J. Rumor cascades [Текст] / A. Friggeri, L.A. Adamic, D. Eckles, J. Cheng. - 2015. – С. 101-110.

6. Harrison M. A Complete Walkthrough of Beginning Python with Unique Illustrations Showing how Python Really Works [Текст] / M. Harrison. - 2017. - №1.

7. Zhang Y., Wang , Luo J. Detection and Analysis [Текст] / Y. Zhang, Wang, J. Luo // Social and Information Networks (cs.SI). - 2016.

8. Jin , Cao , Zhang , Luo. News Verification by Exploiting Conflicting [Текст] / Jin , Cao , Zhang , Luo. // 2016. – С. 2972 – 2978.

9. Jing M., Wei G., Prasenjit M., Sejeong K., Bernard J.J., Wong K.F., Cha. Detecting Rumors from Microblogs with Recurrent Neural Networks [Текст] / M. Jing, G. Wei, M. Prasenjit, K. Sejeong, J.J. Bernard, K.F. Wong// Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. - 2016. - С. 3818-3824.

10. Kirk M. Thoughtful Machine Learning with Python [Текст] / M. Kirk // O'Reilly Media. -2017. №1. – С. 220.

11. Kojouharov. Cheat Sheets for AI, Neural Networks, Machine Learning, Deep Learning & Big Data [Электронный ресурс]. - https:// becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463

12. Kurt T., Chris G., Dawn S., Vern A.P. Suspended Accounts in Retrospect: An Analysis of Twitter Spam [Текст] / T. Kurt, G. Chris, S. Dawn S., A.P.Vern. // Proceedings of the ACM SIGCOMM Conference on Internet Measurement Conference. - 2011. - С. 243-258.

13. Kwon S., Cha M., Jung K., Chen W., Wang Y. Prominent features of rumor propagation in online social media. [Текст] / S. Kwon, M. Cha, K. Jung, W. Chen // International Conference. - 2013. - С. 1103–1108.

14. Lamons M., Nagaraja R.K. Python Deep Learning Projects [Текст]: учебное пособие/ M. Lamons, R.K.Nagaraja // Packt. – 2018. – С. 472.

15. Liu Y. Python Machine Learning By Example [Текст]: учебное пособие / Y.Liu // Packt. – 2017. – С. 254.

16. Lutz M. Learning Python [Текст]: учебное пособие / M. Lutz // O'Reilly Media. -2013. №1. – С. 1213.

17. Martí P., Serrano-Estrada L., Nolasco-Cirugeda A. Social Media data: Challenges, opportunities and limitations in urban studies [Текст] / P.Martí, L. Serrano-Estrada, A. Nolasco-Cirugeda // Computers, Environment and Urban Systems, №74. - 2019. - С. 161-174.

18. Data Visualization with Python: The Complete Guide: набор лекций в видео формате/ Eduonix Learning Solutions. – 2018.

19. pandas.pydata.org/pandas-docs/ [Элекстронный ресурс] /. - http:// pandas.pydata.org/pandas-docs/stable/index.html

20. Paramita M. Extracting Temporal and Causal Relations between Events [Текст] / M. Paramita // 2014. С. 10-17.

21. Qazvinian V., Rosengren E., Redev D., Qiaozhu M. Rumor has it: Identifying Misinformation in Microblogs [Текст] / V. Qazvinian, E. Rosengren, D. Redev, M. Qiaozhu // Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing. Scotland. 2011. С. 1589-1599.

22. Ramalho L. Fluent Python [Текст] / L. Ramalho // O'Reilly Media. - 2015.- С. 765.

23. Simon P. The Visual Organization: Data Visualization, Big Data, and the Quest for Better Decisions [Текст] / P. Simon // Wiley and SAS Business Series – № 1. – 2014.

24. Soroush V. Automatic Detection and Verification of Rumors [Текст] / V. Soroush // Massachusetts: Massachusetts institute of techology, 2015. – С. 147.

25. Sweigart A. Automate the Boring Stuff with Python [Текст] / Sweigart A.//№1. - 2015. – С. 504.

26. Takahashi T., Igata N. Rumor detection on twitter [Текст] / T. Takahashi, N. Igata // Soft Computing and Intelligent Systems (SCIS). - №6. - 2012. - С. 452-457.

27. Weiling C., Zhanga Y., Tong C., Bu S. Unsupervised rumor detection based on users' behaviors using neural networks [Текст] / 2017.

28. Wikipedia contributors. "List of emoticons," Wikipedia, The Free Encyclopedia [Электронный источник] /. https://en.wikipedia.org/wiki/List_of_emoticons

29. Zhao Z., Resnick P., Mei Q. Enquiring Minds: Early Detection of Rumors in Social Media from Enquiry Posts [Текст] / Z. Zhao , P. Resnick , Q. Mei.// IW3C2. Италия, Флоренция. - 2015.

30. Клименко С. Сбор маркетинговой информации и конкурентная разведка с использованием социальных сетей [Текст] / С. Клименко // ФИНАНСОВАЯ ЖИЗНЬ. - 2012. - С. 27-31.

31. Коршунов А., Белобородов И., Бузун Н., Аванесов В. Анализ социальных сетей: методы [Текст] / А. Коршунов, И. Белобородов, Н. Бузун, В. Аванесов // Труды Института системного программирования РАН, Том 26, № 1. - 2014. - С. 439-456.

32. Мартышкин А.И., Сальников И.И., Пащенко Д. Этапы сбора и представления больших данных [Текст] / А.И. Мартышкин, И.И. Сальников, Д. Пащенко // Известия Тульского Государственного Университета, Технические Науки, № 9. – 2018. – С. 617-628.

33. Мильчук Я. Анализ данных социальных сетей как способ сбора информации [Текст] / Я. Мильчук // Научный журнал, Том 5, № 28. – 2018. – С. 30-31.

34. Черняев А. Ивашко А. Метод оценки наличия слухов в сообщениях микроблоговой социальной сети Twitter [Текст] / А. Черняев // Математика и междисциплинарные исследования, материалы Всерос. Науч.-практ. Конф. Молодых ученых с международным участием. Г. Пермь. – 15-18 мая 2019г. – С. 418-422.