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

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Разработка метода прогнозирования распространения публикаций в социальных сетях на основе модели независимых каскадов»

Студент группы ИУ7-82 _____ О.В. Яковлева
(Подпись, дата)

Руководитель ВКР _____ И.В. Рудаков
(Подпись, дата)

Нормоконтролер _____ Ю.В. Строганов
(Подпись, дата)

2019 г.

Реферат

Расчетно-пояснительная записка 79 с., 16 рис., 5 табл., 39 источников, 1 прил.

СОЦИАЛЬНАЯ СЕТЬ, РАСПРОСТРАНЕНИЕ ПУБЛИКАЦИЙ, МОДЕЛЬ НЕЗАВИСИМЫХ КАСКАДОВ, ВЕРОЯТНОСТЬ РАСПРОСТРАНЕНИЯ, АЛГОРИТМ МАКСИМИЗАЦИИ ОЖИДАНИЯ

Объектом исследования является процесс распространения публикаций в ОСС SinaWeibo. Предметом исследования является модель независимых каскадов прогнозирования распространения публикаций в данной ОСС.

Цель работы — разработка метода построения каскада распространения заданной публикации пользователя онлайновой социальной сети с использованием модели независимых каскадов.

Поставленная цель достигается за счет определения параметра модели независимых каскадов – вероятности распространения – как функции атрибутов пользователей, пары связанных пользователей и пары пользователь-публикация и дальнейшего прогнозирования методом моделирования.

Перечень условных обозначений

ОСС – онлайновая социальная сеть

МНК – модель независимых каскадов

АсМНК – асинхронная модель независимых каскадов

Содержание

Введение

Понятие социальной сети впервые было введено в 1954 году социологом Джоном Барнсом и означало «социальную структуру, состоящую из группы узлов, которыми являются социальные объекты (люди или организации), и связей между ними» [1].

Появление первой современной социальной сети обусловлено созданием сайта Эндрю Вейнрейха SixDegrees.com в 1997 году [2]. Ее основатель предложил пользователям создавать личные профили и списки своих друзей, а уже с 1998 года они имели возможность просматривать их страницы.

Сегодня социальная сеть - это веб-сервис, позволяющий пользователям создавать личный аккаунт, хотя бы частично открытый для других людей; управлять списком пользователей, с которыми поддерживается связь; а также просматривать и отслеживать связь других людей [3]. Аккаунт (англ. account – учетная запись, профиль, страница) представляет собой регистрационную запись в социальной сети, которая содержит сведения о пользователе (имя, фамилию, пол, фотографию и прочее) [4]. В собственном профиле люди могут размещать разного рода информацию: начиная с личных данных и заканчивая распространением новостных заметок, записей со страниц других участников сети.

Публикации можно классифицировать на вирусные и обычные. К вирусным публикациям относятся те, которые охватывают большое количество пользователей. Прогнозирование распространения публикаций позволяет понять, станет ли публикация вирусной. Это важная задача в области маркетинга и рекламы.

Целью работы является разработка метода построения каскада распространения заданной публикации пользователя онлайновой социальной сети с использованием модели независимых каскадов.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Рассмотреть существующие методы, использующие модель независимых каскадов или её асинхронное расширение для построения каскадов распространения
2. Разработать алгоритм настройки параметра модели независимых каскадов
3. Реализовать метод построения каскада распространения с использованием модели независимых каскадов в программном обеспечении
4. Провести апробацию метода на наборе данных одной из существующих онлайновых социальных сетей и дать рекомендации о применимости разработанного метода.

1 Аналитический раздел

1.1 Описание предметной области

Формально социальная сеть представляет собой ориентированный граф $G(V, E)$, в котором $V = \{v_1, \dots, v_N\}$ – конечное множество вершин (пользователей) и E – множество ребер, отражающих связи между пользователями. Если $(v_i, v_j) \in E$, пользователь v_j называется *подписчиком* пользователя v_i, а пользователь v_i – *другом* пользователя v_j.

Одним из основных средств информационного взаимодействия между пользователями социальной сети является публикация записей или постов (англ. post – сообщение). Пользователи публикуют записи, чтобы поделиться со своими подписчиками различной информацией, например, отзывами о каких-либо товарах или услугах, политическими взглядами, идеями, мнением и т.д. Каждая запись характеризуется:

- содержанием (текстовая, звуковая, графическая или видеоинформация);
- временем публикации;
- упоминаниями (ссылками на профили пользователей, к которым обращена запись), опционально;
- комментариями (сообщениями, выражающими реакцию других пользователей на запись), опционально;
- оценками “мне нравится” (отметками, выражающими одобрение записи другими пользователями), опционально.

Из-за эффекта *социального влияния* публикации распространяются в социальной сети по принципу *информационного каскада* [5]. В результате этого некоторые *темы* могут приобрести высокую популярность и способствовать появлению новых тенденций.

Социальное влияние – процесс и результат изменения субъектом влияния поведения объекта влияния, его установок, намерений, представлений и оценок, а также основывающихся на них действий в ходе взаимодействия с ним [6]. В

социальных сетях влияние проявляется в осуществлении *перепубликации* сделанной пользователем записи его подписчиками.

Информационный каскад - ситуация, когда индивид, принимая решение, основывается на наблюдаемом поведении других индивидов, уже сделавших выбор [1].

Перепубликация - копия записи, публикуемая пользователем от своего имени, но с сохранением ссылки на первоначального автора [7].

Тема - множество семантически связанных термов, выражающих один предмет обсуждения, например, {“обама”, “посетить”, “китай”} [5].

ER-модель предметной области представлена на рисунке 1.1.

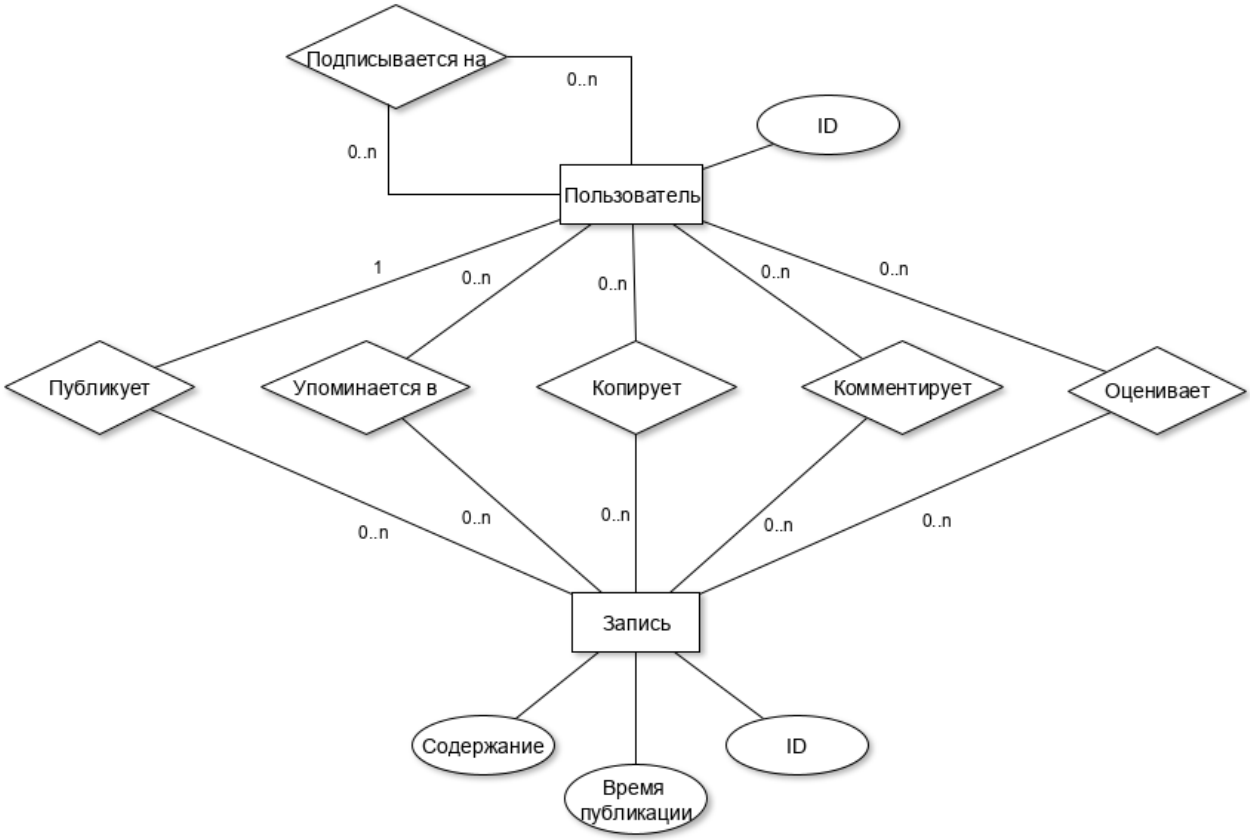

Рисунок 1.1. ER-модель предметной области

1.2 Проблема распространения информации в социальных сетях

Можно выделить четыре основные задачи, связанные с проблемой распространения информации: обнаружение популярных тем, моделирование процесса распространения, определение наиболее влиятельных пользователей, изучение изменения мнений пользователей.

Задача обнаружения популярных тем заключается в разработке средств автоматического обнаружения тем, которые популярны среди пользователей на данный момент времени или станут популярны в будущем [5]. Традиционные методы определения таких тем, разработанные для обработки статических корпусов текстов, не применимы к динамическим потокам записей, генерируемым пользователями онлайновых социальных сетей.

Ю. Лесковец и соавторы [8] показали, что временная динамика популярности темы характеризуется резким ростом интереса пользователей к данной теме до определенного пика и дальнейшим падением.

Таким образом, *популярными* считаются темы, которые вызывают наибольший всплеск интереса пользователей, то есть наиболее резкий рост числа записей в общем потоке.

Существует два основных способа обнаружения всплесков интереса к теме: анализ частоты появления представляющих ее термов и анализ частоты социального взаимодействия. В работе [9] предложен простейший метод обнаружения всплесков, основанный на расчете нормированной частоты появления терма в записях. В работе [10] предлагается подход, основанный на итерационном обновлении тематической модели (модели, позволяющей для каждого документа найти темы, которые его описывают, и кроме того показывающей, какие слова характеризуют ту или иную тему) при появлении новых записей. Для каждой темы составляется эволюционная матрица, отражающая изменение интереса к теме со временем, анализ которой позволяет обнаружить всплески интереса.

В работе [11] для обнаружения популярных тем предлагается использовать социальные аспекты ООС вместо текстового содержания записей. В основе данного подхода лежит анализ ссылок между пользователями, которые создаются автоматически социальной сетью, когда один пользователь оставляет комментарий к записи другого или осуществляет перепубликацию, или вручную пользователем, когда в опубликованной им записи присутствуют упоминания.

Задача моделирования процесса распространения заключается в нахождении структуры каскада распространения и его временной динамики.

Последовательность активации – множество вершин, упорядоченное в зависимости от времени принятия вершиной информации [5].

Каскад распространения – направленное дерево, в корне которого находится первая вершина из последовательности активации, а ребра показывают направление распространения информации (передачу информации от одной вершины к другой) [5].

Процесс распространения информации характеризуется двумя аспектами: его структурой, т.е. графом распространения, описывающим влияние вершин друг на друга, и временной динамикой, т.е. количеством вершин, владеющих информацией, в каждый момент времени.

Можно выделить две категории моделей распространения информации: объяснительные и прогностические.

Цель объяснительных моделей заключается в выводе каскада распространения на основе заданной последовательности активации. Существует несколько алгоритмов построения данного класса моделей.

Гомез и соавторы в [12] предлагают итеративный алгоритм NETINF, который, используя данные о времени принятия вершиной информации и вероятностях влияния вершин на соседние находит каскад распространения, максимизирующий правдоподобие наблюдаемой последовательности активации.

Гомез и соавторы в [13] расширяют алгоритм NETINF и предлагают моделировать процесс распространения информации как дискретную сеть непрерывных, условно независимых временных процессов, происходящих с разными скоростями. Вероятность передачи информации от одной вершины к другой в данный момент времени задается с помощью функции плотности вероятности, которая зависит от времени принятия вершиной информации и скорости передачи между двумя вершинами. Предложенный алгоритм

NETRATE позволяет получить каскад распространения путем формулирования и решения выпуклой задачи максимального правдоподобия [14].

Оба описанных выше алгоритма работают со статическими сетями, что является их недостатком, так как топология онлайновых социальных сетей постоянно изменяется. Этот недостаток Гомез и соавторы устраняют в [15], предлагая алгоритм INFOPATH, который использует стохастический градиентный спуск и позволяет отслеживать изменения путей распространения информации с течением времени.

Прогностические модели нацелены на прогнозирование разворачивания новых каскадов распространения во времени и/или в пространстве путем обучения на предыдущих каскадах. В [5] прогностические модели классифицируются на графовые и неграфовые.

Примером неграфовых моделей являются модели эпидемии. Впервые они были сформулированы в работе [16] в 1921г. Их использование для моделирования процесса распространения информации основано на схожести данного процесса и эпидемии. Двумя основными моделями эпидемии являются SIS и SIR, предполагающие деление всего множества вершин на несколько классов: S – “уязвимые” (англ. “susceptible”), I – “зараженные” (англ. “infected”), R – “выздоровевшие” (англ. “recovered”). В случае модели SIS “выздоровевшие” вершины становятся снова “уязвимыми”, а в случае SIR - остаются в классе “выздоровевших”. В обеих моделях используются два параметра: β – вероятность “заражения”, то есть перехода из класса S в класс I, и γ – вероятность “выздоровления”, то есть перехода из класса I в класс R (для SIR) или S (для SIS). Количество вершин в каждом классе в заданный момент времени описывается системой дифференциальных уравнений.

Модели эпидемии предполагают одинаковые вероятности наличия связей между каждой парой вершин, что делает их нереалистичными для использования в онлайновых социальных сетях, где важна топология графа.

Также к неграфовым можно отнести модель Далея-Кендалла имитации процесса распространения слухов [17] и клеточный автомат [18].

Графовые модели предполагают наличие статически заданного направленного графа социальной сети. Эти модели являются имитационными. Каждая вершина графа может находиться либо в активном, либо в неактивном состоянии, причем для каждой вершины возможен переход из неактивного состояния в активное, но не наоборот. На вершину, находящуюся в неактивном состоянии, оказывают влияние все ее активные соседи. Модели данного класса, в отличие от моделей эпидемии, позволяют получать структуру каскада распространения.

К наиболее известным графовым моделям относятся модель независимых каскадов [19] и линейная модель с порогами [20]. В обеих моделях процесс распространения информации носит дискретный характер и разворачивается итеративно, начиная с некоторого начального множества активных вершин, называемых *ранними последователями*.

Ранние последователи – множество пользователей, которые первыми усваивают информацию, а затем запускают процесс дальнейшего распространения.

Параметрами линейной модели с порогами являются степень влияния для каждого ребра, а также порог влияния для каждой вершины графа. На каждой итерации еще неактивная вершина становится активной, если сумма влияний на нее соседних вершин превышает заданный порог. Порог может быть фиксированным для всех вершин [21], может быть выбран случайно согласно некоторому вероятностному распределению [22], а в общем случае индивидуальные различия обуславливаются опытом пользователя, его убежденностью, личностными чертами, воздействием средств информации, воспринимаемыми затратами [23].

Параметром модели независимых каскадов является вероятность активации, заданная для каждого ребра. Пример графа, в котором каждому ребру сопоставлена вероятность активации, приведен на рисунке 1.2. На каждой итерации активная вершина v может активировать соседнюю вершину w с вероятностью $p_{v,w}$, для этого есть только одна попытка. Если вершине v

удалась активация вершины w, вершина w становится активной на следующей итерации. Процесс продолжается до тех пор, пока не остается возможности для активации вершин. Несколько активных вершин могут независимо пытаться активировать своего неактивного соседа, таким образом, вероятность активации тем больше, чем больше у вершины активных соседей.

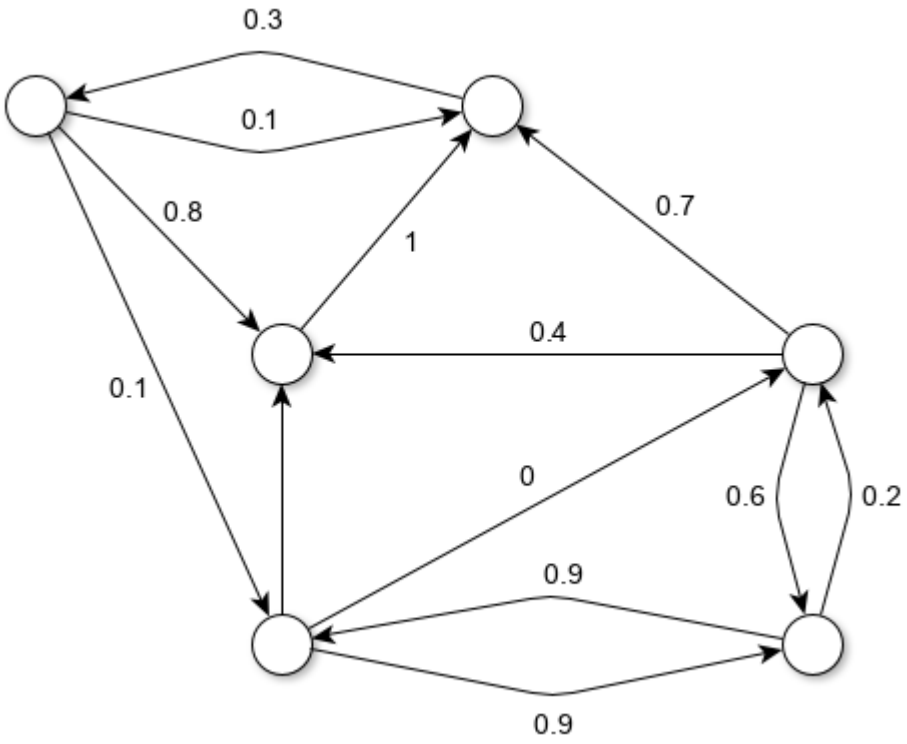

Рисунок 1.2. Пример графа с заданными вероятностями активации

На рисунке 1.3 представлен пример каскада распространения, полученного по модели независимых каскадов.

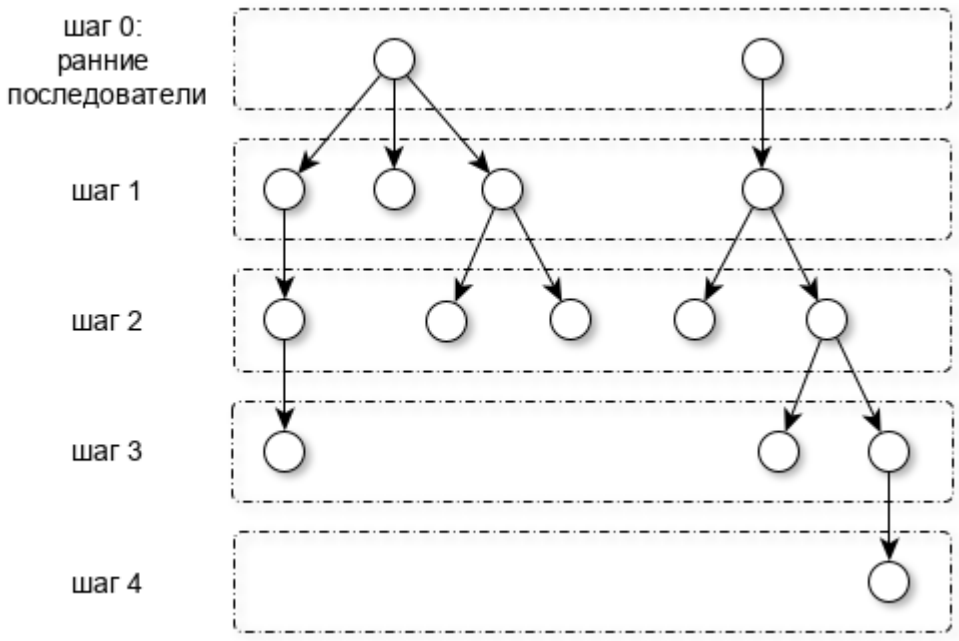

Рисунок 1.3. Каскад распространения, полученный по МНК

Также существуют асинхронные расширения модели независимых каскадов и линейной модели с порогами [24]. В этих моделях процесс распространения информации также разворачивается итеративно, но вдоль непрерывной временной оси. Эти модели используют те же параметры, что и их синхронные аналоги, и дополнительно - параметр задержки распространения для каждого ребра графа.

На рисунке 1.4 представлен пример каскада распространения, полученного по асинхронной модели независимых каскадов.

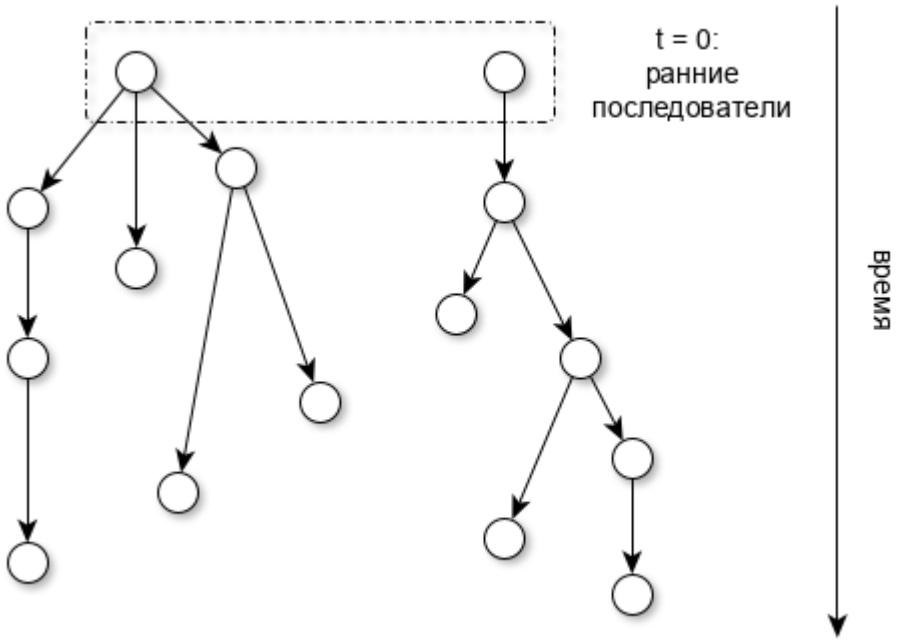

Рисунок 1.4. Каскад распространения, полученный по АсМНК

Задача определения наиболее влиятельных пользователей заключается в нахождении таких пользователей социальной сети, которые, получив информацию, могли бы вызвать наибольшие каскады дальнейших распространений. В работе [25] для оценки распределения влияния вершин социальной сети используется алгоритм PageRank [26]. Рейтинг вершины пропорционален вероятности её посещения в случайном блуждании, где множеством состояний является множество вершин.

Другой класс подходов учитывает признаки вершин и публикаций, а также пути, по которым распространяется информация. В работе [27] предлагается неграфовый, чувствительный к теме публикации метод. Для этого авторы определяют набор признаков вершин и тем публикаций, а затем выполняют кластеризацию. Далее вершины в каждом кластере ранжируются для выявления наиболее влиятельных пользователей для каждой темы.

В работе [28] предлагается использовать модель независимых каскадов и линейную модуль с порогами. Влияние σ(A) множества вершин A авторы [28] определяют как ожидаемое число активных вершин при завершении процесса распространения, инициированного вершинами из множества A. Для обеих моделей (линейного порога и независимых каскадов) возникает NP-трудная

задача: при заданном параметре k найти k-элементное множество A, максимизирующее σ(A). Авторы [28] находят аппроксимирующий алгоритм для решения проблемы максимизации влияния. Поскольку проблема максимизации влияния схожа с известной задачей максимизации субмодулярных функций, то для соответствующего применения алгоритма необходимо лишь доказать, что σ(A) является субмодулярной функцией, что и удалось сделать авторам [28].

Кроме процесса распространения информации также рассматривается процесс формирования и динамики мнений. После получения какого-либо сообщения пользователь социальной сети формирует мнение о нем. Далее мнение пользователей может изменяться с течением времени под влиянием других пользователей. К наиболее известным моделям влияния относятся модель сетевой автокорреляции [29], модель адаптивно-подражательного поведения [30] и марковская модель влияния [31].

На рисунке 1.5 представлена классификация задач, решаемых в области распространения информации в социальных сетях, подходы к их решению и некоторые примеры используемых моделей.

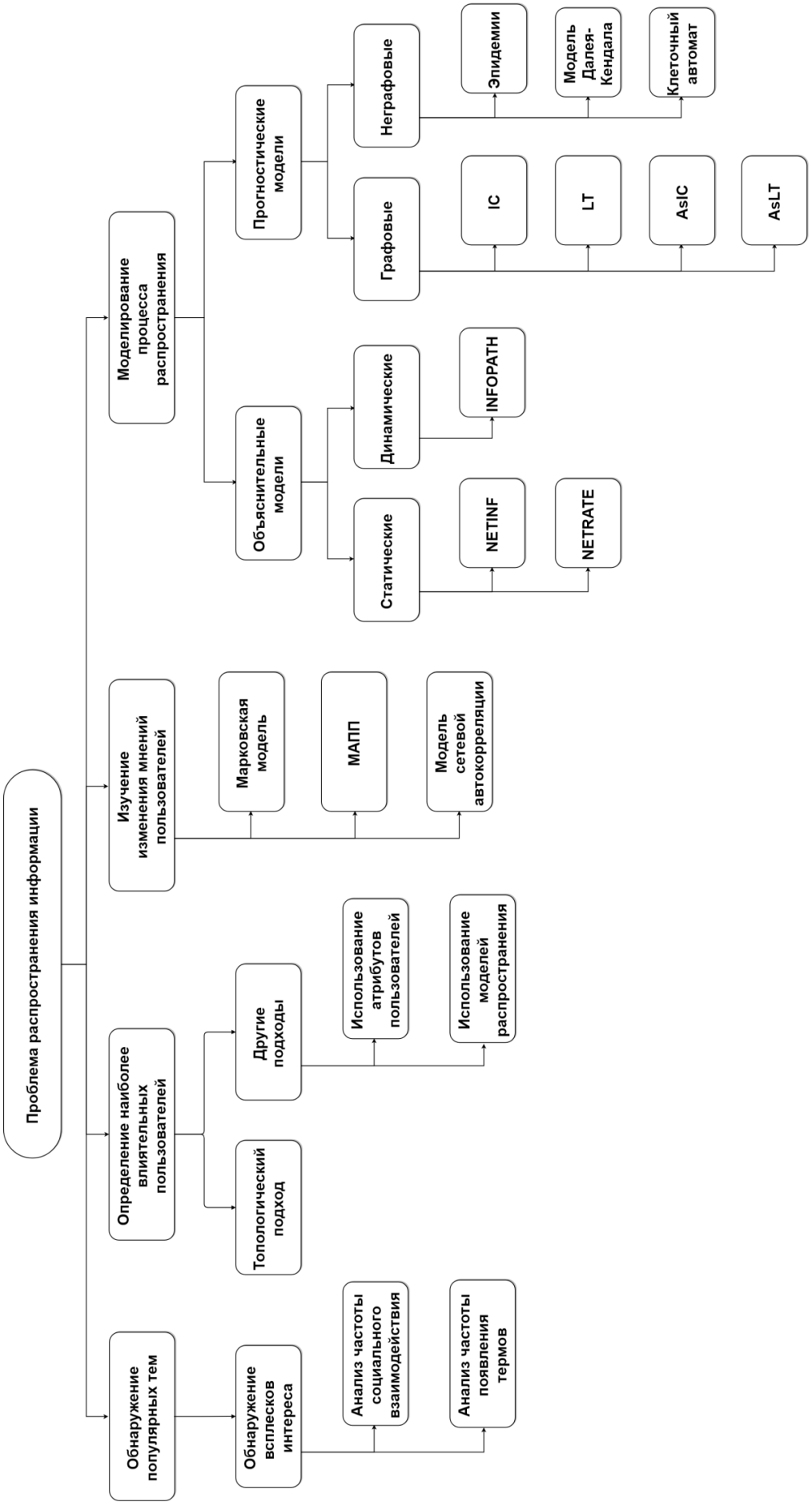

Рисунок 1.5. Задачи в области распространения информации в ОСС

1.3 Постановка задачи прогнозирования по МНК

Основываясь на проведенном анализе предметной области, постановка решаемой в данной работе задачи может быть сформулирована следующим образом.

Пусть $G = (V, E)$ – направленный граф без петель, где V и E $(\subset V \times V)$ – множества вершин и ребер соответственно. Пусть для каждой вершины $v \in V$ определено два множества: $F(v)$ – множество вершин, связанных с v исходящим из v ребром, т.е. $F(v) = \{u \in V; (v, u) \in E\}$, и $B(v)$ – множество вершин, связанных с v входящим в v ребром, т. е. $B(v) = \{u \in V; (u, v) \in E\}$.

Пусть M – множество всех публикаций, $M_u \subset M$ – множество публикаций пользователя $u \in V$, $\mathrm{D}_u \subset M_u$ – множество публикаций пользователя $u \in V$, содержащих упоминания других пользователей из V, $\mathrm{M}_u \subset M$ – множество пользователей, которые упомянуты пользователем $u \in V$ в его публикациях, $t\mathrm{M}^u$ – множество публикаций, в которых упомянут пользователь $u \in V$. Пусть каждая публикация $m \in M$ ассоциирована с множеством K_m ключевых слов.

Процесс распространения публикации $m \in M$ начинается с заданного начального множества активных вершин $D(0)$ и протекает следующим образом. Если вершина v стала активной на шаге t, она получает единственную попытку активировать каждую еще не активную вершину $w \in F(v)$ с вероятностью $p_{v,w}$. Если активация успешна, вершина w становится активной на шаге $\mathrm{t} + 1$, т.е. $w \in D(t + 1)$. Если на шаге t активными стали несколько вершин из множества $B(w)$, они пытаются активировать вершину w последовательно в произвольном порядке, однако все попытки выполняются на шаге t. Вне зависимости от успешности попытки, вершина v не может пытаться активировать вершину w на следующих шагах. Процесс завершается, когда больше нет возможностей активации вершин.

Пусть $D = D(0) \cup D(1) \cup \ldots \cup D(T)$ – эпизод распространения публикации $m \in M$, где $D(t)$ – множество вершин, ставших активными на шаге t. Пусть $C(t)$ – множество узлов, активных к шагу t, т.е. $\mathrm{C(t)} = \bigcup_{\tau \le t} D(\tau)$.

Необходимо реализовать программный модуль, рассчитывающий вероятности $p_{u,v}$ распространения публикации, где $0 \leq p_{u,v} \leq 1$, для каждого ребра $e = (u, v) \in E$. Для работы этого модуля необходимо подготовить обучающую выборку из нескольких каскадов распространения. Пусть $K \subset M$ – множество публикаций, каскады распространения которых используются для обучения модели.

Входные данные модуля:

- $\{D_s : s = 1..S\}$ - множество эпизодов распространения, где $S = |K|$;
- $F(v)$ для каждой вершины $v \in \bigcup_{s=1}^{S} D_s$;
- $B(v)$ для каждой вершины $v \in \bigcup_{s=1}^{S} D_s$;
- M_v для каждой вершины $v \in V$;
- D_v для каждой вершины $v \in V$;
- M_v для каждой вершины $v \in V$;
- $t\mathrm{M}^v$ для каждой вершины $v \in V$;
- K_m для каждой публикации $m \in M$.

Выходным данным является функция P, сопоставляющая каждому ребру $e \in E$ вещественное число от 0 до 1, т.е. $P : E \to [0, 1]$.

Также необходимо разработать модуль, реализующий имитационную модель независимых каскадов для прогнозирования распространения публикации.

Входные данные:

- публикация $m \in M$, распространение которой нужно спрогнозировать;
- $D(0)$ – начальное множество активных вершин;
- G – весь граф социальной сети;
- T_{max}– максимальное количество шагов;
- $P : E \to [0, 1]$.

Выходным данным является каскад распространения публикации m.

Ограничения, в рамках которых будет решена задача:

- начальное множество активных вершин *D(0)* состоит только из одной вершины – пользователя, являющегося автором публикации – для любой публикации $m \in M$;
- максимальное количество шагов (глубина каскада распространения) ограничено входным параметром T_{max};
- распространением считается только перепубликация записи, комментарии и оценки “мне нравится” не учитываются.

Также стоит отметить, что ограничением самой модели независимых каскадов является то, что пользователь u может получить информацию от пользователя v только в том случае, если u является подписчиком v, т.е. $\exists\, e: e(u, v) \in E$.

1.4 Методы расчета вероятностей распространения информации в МНК

Определение вероятностей распространения информации для реальной социальной сети является довольно сложной задачей. Кроме того, вызывает трудности и тестирование таких методов, так как фактические значения вероятностей неизвестны.

Рассмотрены три существующих метода определения вероятностей: метод Сайто для МНК [32], метод Сайто [33] для АсМНК, метод Гиля [34]. Для каждого метода проанализированы достоинства и недостатки.

1.4.1. Метод Сайто для МНК

Сайто и соавторы в работе [32] определяют вероятность того, что вершина w станет активной на шаге $t + 1$ в эпизоде D_s (1) и объективную функцию для S каскадов (2), а затем сводят задачу нахождения вероятности распространения для каждого ребра графа к задаче максимизации этой функции.

$$P_w^{(s)}(t+1) = 1 - \prod_{\mathrm{v}\in \mathrm{B}(\mathrm{w})\cap D_s(\mathrm{t})} (1 - p_{v,w}) \tag{1}$$

$$L(\boldsymbol{\theta}) = \sum_{s=1}^{S} logL(\boldsymbol{\theta}; D_s)$$

$$= \sum_{s=1}^{S} \sum_{t=0}^{T_s-1} \left(\sum_{w\in D_s(t+1)} log(P_w^{(s)}(t+1)) + \right. \tag{2}$$

$$\left. + \sum_{v\in D_s(t)} \sum_{w\in \mathrm{F}(\mathrm{v})\setminus C_s(\mathrm{t}+1)} \log(1 - p_{v,w}) \right)$$

где $\boldsymbol{\theta} = \{p_{v,w} : (v,w) \in \mathrm{E}\}$.

Поскольку аналитически максимизировать предложенную функцию невозможно, авторы используют алгоритм максимизации ожидания.

Для любой пары вершин *(v, w)*, такой, что $v \in D_s$, $w \in D_s$ и $e = (v,w) \in E$ справедливо утверждать, что если $v \in D_s(t)$, то была выполнена попытка активации вершиной v вершины w на шаге t. Если $w \in D_s(t+1)$, то эта попытка завершилась успехом с вероятностью $\frac{\hat{p}_{v,w}}{\hat{P}_w^{(s)}}$ и неудачей с вероятностью $1 - \frac{\hat{p}_{v,w}}{\hat{P}_w^{(s)}}$. Здесь $\hat{p}_{v,w}$ – текущая оценка вероятности $p_{v,w}$, $\hat{P}_w^{(s)}$ вычисляется из (1), а текущие оценки $\widehat{\boldsymbol{\theta}} = \{\hat{p}_{v,w}\}$. Также очевидно, что в случае, если $w \notin C_s(t+1)$, то попытка активации через ребро *e* была неудачной. Учитывая эти случаи, авторы [32] определяют следующую *Q*-функцию для *S* эпизодов (3).

$$Q(\boldsymbol{\theta}|\widehat{\boldsymbol{\theta}}) = \sum_{s=1}^{S}\sum_{t=0}^{T_s-1}\sum_{v\in D_s(t)}\left(\sum_{w\in \mathrm{F(v)}\cap D_s(t+1)}\left(\frac{\hat{p}_{v,w}}{\hat{P}_w^{(s)}} log\ p_{v,w} + \left(1-\frac{\hat{p}_{v,w}}{\hat{P}_w^{(s)}}\right)\log\left(1 - p_{v,w}\right)\right) + \sum_{w\in \mathrm{F(v)}\setminus C_s(\mathrm{t+1})}\log(1-p_{v,w})\right) \quad (3)$$

Новые оценки $p_{v,w}(5)$ можно найти из условия оптимальности (4):

$$\frac{\partial Q}{\partial p_{v,w}} = 0 \quad (4)$$

$$p_{v,w} = \frac{1}{\left|S_{v,w}^{+}\right| + \left|S_{v,w}^{-}\right|}\sum_{s\in S_{v,w}^{+}}\frac{\hat{p}_{v,w}}{\hat{P}_w^{(s)}} \quad (5)$$

где $S_{v,w}^{+} = \{D_s: (\exists \mathrm{t}\ \epsilon\ [0..T_s-1])\ v\ \epsilon\ D_s(t), w\ \epsilon\ D_s(\mathrm{t}+1)\}$,

$S_{v,w}^{-} = \{D_s: (\exists \mathrm{t}\ \epsilon\ [0..T_s-1])\ v\ \epsilon\ D_s(t), w\ \notin\ D_s(\mathrm{t}+1)\}$

Таким образом, метод состоит из двух повторяющихся шагов:

- оценить текущие вероятности успешной активации $\hat{P}_w^{(s)}$ из (1);
- обновить вероятности распространения информации $p_{v,w}$ из (5).

Достоинство данного метода состоит в относительной простоте его реализации, так как для расчета вероятностей распространения необходима только обучающая выборка из нескольких каскадов в рассматриваемом графе. Недостаток метода заключается в необходимости проведения большого количества вычислений, так как графы реальных социальных сетей могут содержать миллионы вершин и миллиарды связей [35]. Обойти этот недостаток можно, если рассматривать не весь граф целиком, а только некоторый подграф. Но в этом случае прогнозировать распространение новых каскадов можно будет только для того подграфа, на котором обучалась модель.

1.4.2. Метод Сайто для AcMHK

Работа [33] посвящена методу нахождения параметров асинхронной модели независимых каскадов. Сайто и соавторы в [33] рассматривают вероятность распространения (6) и временную задержку (7) как функции *J*-размерного вектора атрибутов вершин.

$$\mathrm{p}_{\mathrm{v,w}} = \mathrm{p}\left(\mathbf{x}_{\mathrm{v,w}}, \boldsymbol{\theta}\right) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}_{\mathrm{v,w}})} \tag{6}$$

$$\mathrm{r}_{\mathrm{v,w}} = \mathrm{r}\left(\mathbf{x}_{\mathrm{v,w}}, \boldsymbol{\varphi}\right) = \exp(\boldsymbol{\varphi}^{\mathrm{T}}\mathbf{x}_{\mathrm{v,w}}) \tag{7}$$

где $\boldsymbol{x}_{v,w}$ – вектор размерности *J+1*, каждый элемент которого равен значению некоторой функции *j*-го атрибута вершин, т.е. $x_{v,w,j} = f_j\left(v_j, w_j\right), j\,\epsilon[1..J];\ x_{v,w,0} = 1;$

$\boldsymbol{\theta}^T = (\theta_0, \dots, \theta_J)$,

$\boldsymbol{\varphi}^T = (\varphi_0, \dots, \varphi_J)$ – транспонированные векторы параметров, причем $\theta_0 = const, \varphi_0 = const.$

Аналогично работе [32] авторы [33] вводят объективную функцию и предлагают максимизировать её, используя итеративный алгоритм максимизации ожидания. Достоинство метода состоит в том, что модель можно обучить на небольшом подграфе $G' \subset G$, а затем использовать её для прогнозирования распространения новых каскадов во всем графе *G*. Недостаток модели заключается в трудоемкости её обучения, так как текущие оценки $\widehat{\boldsymbol{\theta}}$ и $\widehat{\boldsymbol{\varphi}}$ на каждой итерации нельзя найти аналитически и для их нахождения авторы [33] предлагают использовать метод Ньютона. Кроме того, авторы [33] тестируют модель на синтетически сгенерированных атрибутах вершин и не делают никаких предположений о том, что может выступать в качестве атрибутов в реальных социальных сетях. Таким образом, данная модель не применима на практике.

1.4.3. Метод Гиля

Работа [34] также посвящена асинхронной модели независимых каскадов. Гиль и соавторы в [34] рассматривают три группы признаков, которые, по их мнению, влияют на вероятность распространения информации от одной вершины к другой: социальную, семантическую и временную. К социальной группе относятся следующие пять признаков:

- активность (I) – признак, характеризующий относительную активность пользователя, определяется как среднее количество публикаций в час (8).

$$I(u) = \begin{cases} \frac{|M_u|}{\epsilon}, & \text{если } |M_u| < \epsilon \\ 1, & \text{иначе} \end{cases} \quad (8)$$

где ϵ = 30.4 * 24;

- социальная однородность (H) – признак, характеризующий похожесть множеств пользователей, обсуждаемых пользователями u_1 и u_2, определяется с использованием коэффициента Жаккара (9).

$$H(u_1, u_2) = \frac{|\mathrm{M}_{u_1} \cap \mathrm{M}_{u_2}|}{|\mathrm{M}_{u_1} \cup \mathrm{M}_{u_2}|} \quad (9)$$

- частота публикаций с упоминаниями (dTR) – признак, характеризующий склонность пользователя публиковать записи для других пользователей, определяется как отношение количества публикаций, содержащих упоминания, к общему количеству публикаций (10).

$$dTR(u) = \begin{cases} \frac{|D_u|}{|M_u|}, & \text{если } |M_u| > 0 \\ 0, & \text{иначе} \end{cases} \quad (10)$$

- флаг наличия упоминаний одного пользователя в публикациях другого (hM) – признак, характеризующий существование социальной взаимосвязи между пользователями:

$$hM(u_1, u_2) = \begin{cases} 1, & \text{если } u_2 \in \mathrm{M}_{u_1} \\ 0, & \text{иначе} \end{cases} \quad (11)$$

- частота упоминаний (mR) – признак, характеризующий популярность пользователя, определяется количеством публикаций, в которых он упомянут (12).

$$mR(u) = \begin{cases} \frac{|t\mathrm{M}^u|}{\mu}, & \text{если } |t\mathrm{M}^u| < \mu \\ 1, & \text{иначе} \end{cases} \quad (12)$$

где μ выбирается исходя из наблюдаемого распределения частот упоминаний.

К семантической группе относится один признак: флаг наличия хотя бы одного ключевого слова распространяемой публикации в предыдущих публикациях пользователя (hK) – признак, характеризующий интерес пользователя к теме публикации (13).

$$hK(u, m) = \begin{cases} 1, & \text{если } K_m \cap K_u \neq \varnothing \\ 0, & \text{иначе} \end{cases} \quad (13)$$

где $K_u = \bigcup_{m \epsilon M_u} K_m$

К временной группе относится также один признак. Для его расчета сутки делятся на 6 периодов по 4 часа, и заполняется шестимерный вектор V, элементы которого содержат нормированное количество публикаций, сделанных пользователем за соответствующий период: $\sum_{i=0}^{5} V^i = 1$. Данный признак позволяет получить долю активности в течение суток t для каждого пользователя (14).

$$A(u,t) = V_u^{t'} \tag{14}$$

где $t' = \left\lfloor \frac{t}{4} \right\rfloor$

Затем авторы [34] предлагают рассматривать задачу нахождения вероятностей распространения как задачу классификации. Составляется обучающая выборка, в которой в качестве входных переменных выступают описанные выше признаки, а в качестве выходной – факт совершения пользователем перепубликации. То есть, если активация пользователя w пользователем v прошла успешно, выходная переменная принимает значение 1, иначе 0. Таким образом, вероятность распространения определяется как вероятность принадлежности ребра $e = (v, w)$ к классу “успешная активация”. Сама задача классификации решается с использованием логистической регрессии.

Достоинства метода:

- простота и высокая скорость обучения модели, поскольку нет необходимости рассматривать все попытки успешных и не успешных активаций, достаточно только части примеров;
- высокая точность прогнозирования, подтвержденная практическими экспериментами.

Недостатки метода:

- для каждого каскада нужно заново рассчитывать вероятности активации всех ребер, так как среди признаков есть семантический;
- недостаточное внимание уделено выбору признаков и оценке их влияния на выходную переменную.

В таблице 1.1 приведены результаты сравнения описанных выше методов, при условии, что для обучения модели будет использован некоторый небольшой подграф $G' \subset G$.

Таблица 1.1 - Сравнение методов нахождения вероятностей распространения

Метод	для обучения достаточно части примеров успешных и неудачных активаций вершин	учет атрибутов вершин	учет атрибутов связей	учет атрибутов публикаций	предложены конкретные атрибуты	прогнозирование для любого подграфа
Сайто для МНК	–	–	–	–	–	–
Сайто для АсМНК	–	+	–	–	–	+
Гиля	+	+	+	+	+	+

1.5 Выбор социальной сети

ОСС, выбираемая для исследования процесса распространения публикаций, должна удовлетворять следующим критериям:

- возможность сделать перепубликацию записи;
- наличие API, позволяющего получить необходимые для решения задачи, поставленной в п.1.3, данные: список связей пользователя, информацию о публикациях, включая текст, дату, автора;
- ограничивать количество символов в тексте публикаций;
- иметь популярность у пользователей.

На диаграмме, подготовленной аналитическим агентством Statista [35], представлено количество активных пользователей (в миллионах) в самых популярных социальных сетях мира: Facebook, YouTube, Instagram, QZone, Tumblr, SinaWeibo, Twitter.

Для этих социальных сетей составлена таблица соответствия перечисленным выше критериям (таблица 1.2)

Таблица 1.2 - Сравнение ОСС

ОСС	механизм перепубликации	наличие API	ограничение количества символов	количество пользователей (млн)
Facebook	+	+	–	2061
YouTube	–	+	–	1500
Instagram	–	+	–	700
QZone	+	–	–	606
Tumblr	+	+	–	368
SinaWeibo	+	+	+	361
Twitter	+	+	+	328

Сравнительный анализ наиболее популярных ОСС показывает, что выдвинутым требованиям удовлетворяют SinaWeibo и Twitter. Однако API Twitter является закрытым, и для доступа к нему требуется заполнить заявку и получить подтверждение [36]. С другой стороны, API SinaWeibo может использовать любой желающий после регистрации [37].

Таким образом, в качестве объекта исследования выбран процесс распространения публикаций в ОСС SinaWeibo. Предметом исследования является модель независимых каскадов прогнозирования распространения публикаций в данной ОСС.

1.6 Выбор набора данных

Поскольку сбор и обработка данных с помощью API требует значительного количества времени и вычислительных ресурсов, в данной работе будет использоваться один из наборов данных, подготовленный другими исследователями.

Выбранный набор должен содержать данные, необходимые для решения задачи, поставленной в п.1.3. Этому критерию удовлетворяет набор данных

Weibo-Net-Tweet [38], собранный группой китайских исследователей, изучавших влияние друзей пользователя на его поведение [39].

Набор данных получен следующим образом. В список отслеживаемых пользователей добавлены 100 случайных пользователей и все их подписчики. Для каждого пользователя из этого списка ежедневно в течение месяца (с 28.09.2012 по 29.10.2012) осуществлялся сбор его подписчиков. В результате собрано 1,8 миллионов пользователей и 413,5 миллионов связей между ними, в среднем 230 связей у каждого пользователя. Для каждого пользователя собрано 1000 последних публикаций (включая как оригинальные публикации, так и перепубликации). В сумме получено 1 миллиард публикаций.

Далее авторы [39] выбрали 300.000 популярных эпизодов распространения публикаций из всего набора данных. Каждый эпизод распространения содержит оригинальную публикацию и все её перепубликации. В среднем каждая публикация была скопирована около 125 раз. Статистика полученного набора данных приведена в таблице 1.3.

Таблица 1.3 - Статистика набора данных

Набор данных	Количество пользователей	Количество связей	Количество оригинальных публикаций	Количество перепубликаций
Weibo-Net-Tweet	1.787.443	413.503.687	300.000	37.372.573

Набор данных состоит из 8 частей, в данной работе будут использованы только 3:

- файл связей пользователей;
- файл эпизодов распространения;
- файл содержания оригинальных публикаций.

Файл связей пользователей содержит строки в следующем формате:

id1 id2 t

Каждая строка означает, что пользователь с идентификатором id1 является подписчиком пользователя с идентификатором id2 в отметке времени t. Временная отметка соответствует количеству дней, прошедших с начала сбора данных.

Файл эпизодов распространения содержит пары строк в следующем формате:

original_mid original_time original_uid retweet_num

retweet_uid retweet_time retweet_uid retweet_time…

Этой парой строк описывается один эпизод распространения. Первая строка пары содержит идентификатор оригинальной публикации (*original_mid),* дату (*original_time*), идентификатор автора (*original_uid*) и количество перепубликаций (*retweet_num*). Вторая строка содержит все перепубликации, каждая из которых описывается идентификатором пользователя, осуществившего перепубликацию (*retweet_uid*), и дату (*retweet_time*).

Файл содержания оригинальных публикаций содержит наборы из 2-4 строк в следующем формате:

original_mid

content

[@]id1 id2…

[link]

Первая строка набора – идентификатор оригинальной публикации. Вторая строка содержит хешированные значимые слова публикации (хеширование осуществлялось заменой слова на его уникальный числовой идентификатор). Третья строка является опциональной и содержит символ @, за которым следуют идентификаторы пользователей, упомянутых в публикации, разделенные пробелами. Четвертая строка также опциональна, в ней перечисляются url внешних ссылок, присутствующих в тексте публикации.

1.7 Вывод

В данном разделе проведена формализация предметной области, классифицированы задачи, связанные с распространением информации в социальных сетях. Сформулирована задача прогнозирования распространения публикаций и рассмотрены существующие методы её решения, использующие модель независимых каскадов. По результатам проведенного анализа принято решение разработать собственный алгоритм нахождения функции вероятности распространения публикации, совмещающий достоинства метода Сайто для АсМНК [33] и метода Гиля [34]. Также выбрана социальная сеть и набор данных для исследования.

2 Конструкторский раздел

2.1 Метод определения вероятностей распространения

Аналогично [33], вероятность распространения будет задана как функция некоторых атрибутов. Однако, в отличие от [33], для расчета вероятностей будут использованы не только атрибуты пользователей, но и атрибуты связей и публикаций.

Пусть $A(v)$ – множество атрибутов вершины $v \in V$, $A(e)$ – множество атрибутов ребра $e = (v, w) \in E$, $A(m)$ – множество атрибутов публикации m. Пусть $p_{v,w,m}$ – вероятность распространения публикации m от вершины v к вершине w через ребро $e = (v, w) \in E$. Исходя из предположения, что $p_{v,w,m}$ зависит от значений, которые принимают атрибуты из множества $A = A(v) \cup A(w) \cup A(e) \cup A(m)$, $p_{v,w,m}$ определяется следующим образом (15).

$$p_{v,w,m} = p(\boldsymbol{x}_{v,w,m}, \boldsymbol{\theta}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \boldsymbol{x}_{v,w,m})} \quad (15)$$

где $\boldsymbol{x}_{v,w,m} = (a_0, a_1, \dots, a_J)^T$ – вектор значений атрибутов $a_k \in A, k = 1..J, J = |A|, a_0 = 1$,

$\boldsymbol{\theta}^T = (\theta_0, \dots, \theta_J)$ – транспонированный вектор параметров, причем $\theta_0 = const$.

Таким образом, вероятности распространения однозначно определяются значениями атрибутов вершин, ребер и публикации, а также вектором параметров $\boldsymbol{\theta}$.

Аналогично [32] (см.п.1.4.1), вероятность того, что w станет активной на шаге $t + 1$, рассчитывается по формуле (16). Отличие от [32] состоит в том, что здесь вероятность является функцией.

$$P_w(t+1) = 1 - \prod_{\mathrm{v}\in\mathrm{B(w)}\cap\mathrm{D(t)}} (1 - p(\boldsymbol{x}_{v,w,m}, \boldsymbol{\theta})) \quad (16)$$

Эта формула получена из следующих соображений. Вершина не станет активной, если ни одна из соседних активных вершин не сможет её активировать. Вершина станет активной, если событие неудачной активации не наступит.

Аналогично [32] (см.п.1.4.1) для эпизода D можно определить функцию правдоподобия (17).

$$L(\boldsymbol{\theta}; D) = \quad (17)$$

$$= \left(\prod_{t=0}^{T-1} \prod_{w\in\mathrm{D(t+1)}} P_w(t+1) \right) \left(\prod_{t=0}^{T-1} \prod_{v\in\mathrm{D(t)}} \prod_{w\in\mathrm{F(v)}\setminus\mathrm{C(t+1)}} (1 - p(\boldsymbol{x}_{v,w,m}, \boldsymbol{\theta})) \right)$$

Правдоподобие здесь состоит в следующем: все активные вершины будут активированы (первая скобка в (17)) и все неактивные вершины не будут активированы на соответствующем шаге (вторая скобка в (17)).

Объективная функция в отношении $\boldsymbol{\theta}$ (18) определяется также аналогично [32] (см. п. 1.4.1).

$$L(\boldsymbol{\theta}) = \sum_{s=1}^{S} logL(\boldsymbol{\theta}; D_s) \quad (18)$$

$$= \sum_{s=1}^{S} \sum_{t=0}^{T_s-1} \left(\sum_{w\in D_s(t+1)} log(P_w^{(s)}(t+1)) + \right.$$

$$\left. + \sum_{v\in D_s(t)} \sum_{w\in\mathrm{F(v)}\setminus C_s(\mathrm{t+1})} \log(1 - p(\boldsymbol{x}_{v,w,m}, \boldsymbol{\theta})) \right)$$

$$P_w^{(s)}(t+1) = 1 - \prod_{\mathrm{v}\in \mathrm{B(w)}\cap D_s(\mathrm{t})} (1 - p(\boldsymbol{x}_{v,w,m}, \boldsymbol{\theta})) \tag{19}$$

Здесь $P_w^{(s)}(t+1)$ - вероятность того, что w станет активной на шаге $t+1$ в эпизоде D_s (19).

Требуется найти такие значения вектора параметров $\boldsymbol{\theta}$, которые максимизируют (18). Аналогично [32] (см. п. 1.4.1) определяется Q-функция (20).

$$\begin{aligned} Q(\boldsymbol{\theta}|\widehat{\boldsymbol{\theta}}) = \sum_{s=1}^{S} \sum_{t=0}^{T_s-1} \sum_{v\in D_s(t)} \Bigg(& \sum_{w\in \mathrm{F(v)}\cap D_s(t+1)} \Big(\frac{\hat{p}_{v,w,m}}{\hat{P}_w^{(s)}} \log p(\boldsymbol{x}_{v,w,m}, \boldsymbol{\theta}) + (1 \\ & - \frac{\hat{p}_{v,w,m}}{\hat{P}_w^{(s)}}) \log (1 - p(\boldsymbol{x}_{v,w,m}, \boldsymbol{\theta})) \Big) \\ & + \sum_{w\in \mathrm{F(v)}\setminus C_s(\mathrm{t+1})} \log(1 - p(\boldsymbol{x}_{v,w,m}, \boldsymbol{\theta})) \Bigg) \end{aligned} \tag{20}$$

где $\hat{p}_{v,w,m}$ – текущая оценка вероятности $p(\boldsymbol{x}_{v,w,m}, \widehat{\boldsymbol{\theta}})$,

$\hat{P}_w^{(s)}$ вычисляется из (19).

Для нахождения $\boldsymbol{\theta}$ требуется решить систему уравнений (21).

$$\frac{\partial Q(\boldsymbol{\theta}|\widehat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} = 0 \tag{21}$$

Градиент функции Q выглядит следующим образом (22).

$$\frac{\partial Q(\boldsymbol{\theta}|\widehat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} = \sum_{s=1}^{S}\sum_{t=0}^{T_s-1}\sum_{v\in D_s(t)}\left(\sum_{w\in \mathrm{F(v)}\cap D_s(t+1)}((\frac{\hat{p}_{v,w,m}}{\hat{P}_w^{(s)}} - p(\boldsymbol{x}_{v,w,m},\boldsymbol{\theta}))\boldsymbol{x}_{v,w,m}) - \sum_{w\in \mathrm{F(v)}\setminus C_s(\mathrm{t+1})} p(\boldsymbol{x}_{v,w,m},\boldsymbol{\theta})\boldsymbol{x}_{v,w,m})\right) \quad (22)$$

Так как матрица Гессе (22) отрицательно определена, систему уравнений (21) можно решить, используя метод Ньютона.

$$\frac{\partial^2 Q(\boldsymbol{\theta}|\widehat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}^T} = -\sum_{s=1}^{S}\sum_{t=0}^{T_s-1}\sum_{v\in D_s(t)}\left(\sum_{w\in \mathrm{F(v)}\cap D_s(t+1)}\zeta_{v,w,m}\boldsymbol{x}_{v,w,m}\boldsymbol{x}_{v,w,m}{}^T + \sum_{w\in \mathrm{F(v)}\setminus C_s(\mathrm{t+1})}\zeta_{v,w,m}\boldsymbol{x}_{v,w,m}\boldsymbol{x}_{v,w,m}{}^T\right) \quad (22)$$

где $\zeta_{v,w,m} = p(\boldsymbol{x}_{v,w,m},\boldsymbol{\theta})(1 - p(\boldsymbol{x}_{v,w,m},\boldsymbol{\theta}))$.

2.2 Метод прогнозирования распространения публикаций

На рисунке 2.1 в виде IDEF-0 диаграммы представлены этапы работы разработанного метода. Первым этапом является формирование списка атрибутов, значения которых влияют на вероятность распространения. Второй этап - обучение модели (настройка параметров функции вероятности распространения). Третий этап - построение каскада распространения по настроенной модели независимых каскадов.

Далее будут подробно рассмотрены входные и выходные данные и алгоритм работы каждого из этапов.

2.2.1. Формирование списка атрибутов

Для возможности практического применения разрабатываемого метода необходимо предложить конкретные атрибуты и способы их расчета, как было сделано в [34].

Цель данного этапа – составить список атрибутов вершин, ребер и публикаций, значения которых предположительно могут повлиять на вероятность распространения и которые можно посчитать, используя имеющийся набор данных.

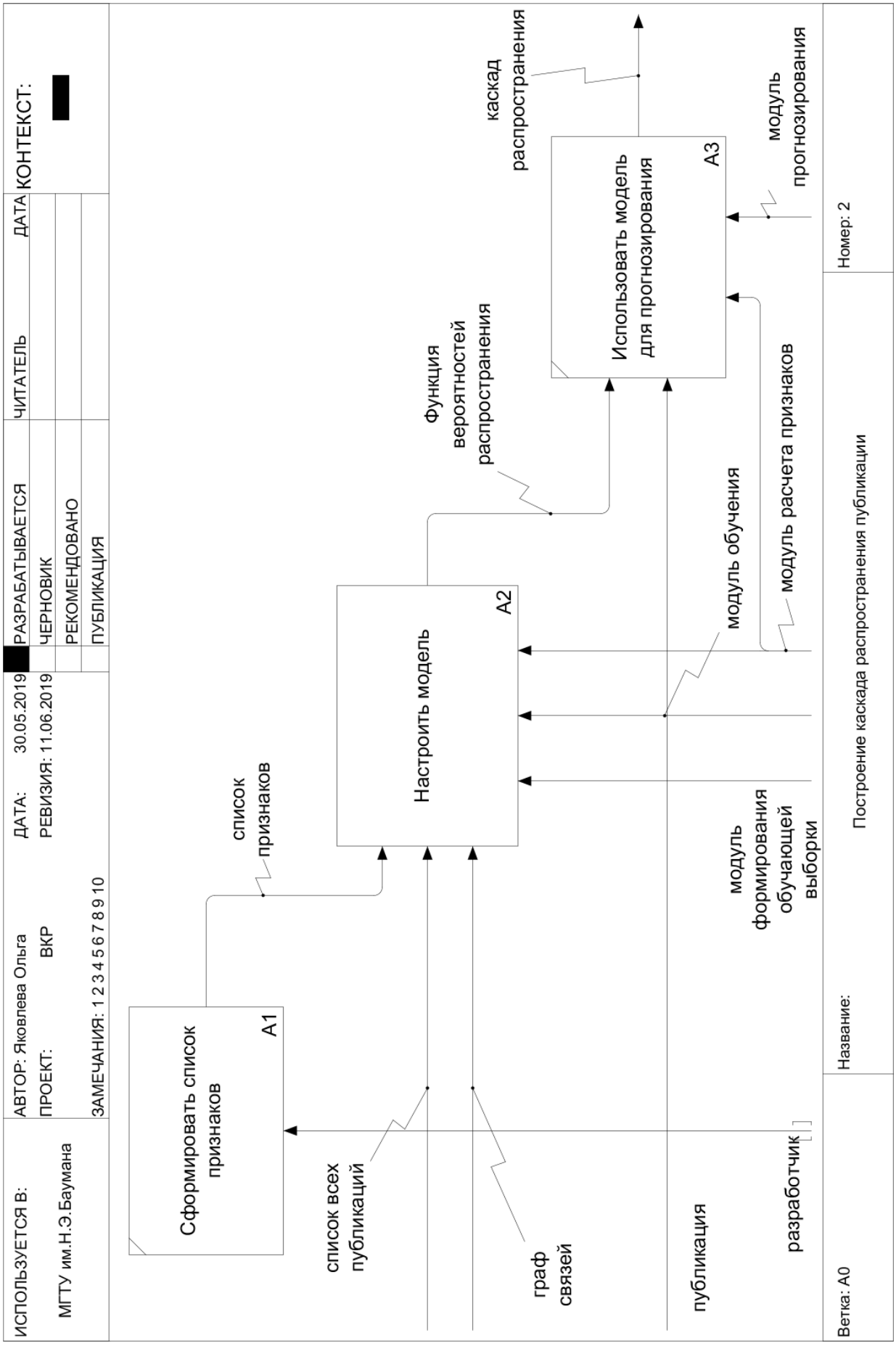

Рисунок 2.1- IDEF0-диаграмма метода прогнозирования

Предлагается выделить следующие атрибуты:

- атрибуты вершин

1) активность (I) – признак, характеризующий относительную активность пользователя, определяется как среднее количество публикаций в час (23).

$$I(u) = \frac{|M_u|}{\epsilon} \tag{23}$$

где $\epsilon = 24$;

2) количество подписчиков (cF) – признак, характеризующий популярность пользователя (24).

$$cF(u) = |F(u)| \tag{24}$$

3) количество друзей (cFr) – признак, характеризующий популярность пользователя (25).

$$cFr(u) = |B(u)| \tag{25}$$

4) количество упоминаний пользователя (cR) – признак, характеризующий популярность пользователя (26).

$$cR(u) = |t\mathrm{M}^u| \tag{26}$$

5) частота публикаций с упоминаниями (dTR) – без изменений относительно метода Гиля (см.п.1.4.3).

- признаки ребер:

1) однородность упоминаний (H) – без изменений относительно метода Гиля (см.п.1.4.3).

2) однородность публикаций (*HP*) – признак, характеризующий похожесть множеств публикаций двух пользователей, определяется с использованием коэффициента Жаккара (27), при этом публикации m1 и m2 считаются одинаковыми, если m1 – перепубликация m2, или m2 – перепубликация m, или m1 и m2 обе являются перепубликациями m3.

$$HP(u_1, u_2) = \frac{|M_{u_1} \cap M_{u_2}|}{|M_{u_1} \cup M_{u_2}|} \tag{27}$$

3) однородность по подписчикам (*HF*) – признак, характеризующий похожесть множеств подписчиков двух пользователей, определяется с использованием коэффициента Жаккара (28).

$$HF(u_1, u_2) = \frac{|F(u_1) \cap F(u_2)|}{|F(u_1) \cup F(u_2)|} \tag{28}$$

4) однородность по друзьям (*HFr*) – признак, характеризующий похожесть множеств друзей двух пользователей, определяется с использованием коэффициента Жаккара (29).

$$HFr(u_1, u_2) = \frac{|B(u_1) \cap B(u_2)|}{|B(u_1) \cup B(u_2)|} \tag{29}$$

5) частота упоминаний одного пользователя в публикациях другого (*rM*) – признак, характеризующий существование социальной взаимосвязи между пользователями (30).

$$rM(u_1, u_2) = \begin{cases} \frac{|\mathrm{M}_{u_1} \cap t\mathrm{M}^{u_2}|}{|\mathrm{M}_{u_1}|}, & \text{если } \left|\mathrm{M}_{u_1}\right| > 0 \\ 0, & \text{иначе} \end{cases} \tag{30}$$

- признак публикации:

  1) частота присутствия хотя бы одного ключевого слова распространяемой публикации в предыдущих публикациях пользователя (rK) – признак, характеризующий интерес пользователя к теме публикации (31).

$$hK(u,m) = \begin{cases} \frac{|\{m: K_m \cap K_u \neq \varnothing\}|}{|M_u|}, & \text{если } |M_u| > 0 \\ 0, & \text{иначе} \end{cases} \quad (31)$$

где $K_u = \bigcup_{m \epsilon M_u} K_m$

Таким образом, три признака, предложенных в работе [34], взято без изменений, все признаки логического типа заменены на соответствующие частотные признаки, частота упоминаний пользователя заменена на количество, изменена формула расчета активности; добавлено пять новых признаков.

2.2.2. Настройка модели

Цель данного этапа – вычислить значения вектора коэффициентов $\boldsymbol{\theta}$.

Основные шаги этапа настройки представлены на рисунке 2.2.

Рисунок 2.2 – IDEF0-диаграмма этапа обучения

Первым шагом обучения является формирование обучающей выборки. Выборка представляет собой матрицу, каждая строка которой имеет следующий вид:

[post_id user1 user2 level result feature1 feature2 … featureN]

где post_id – идентификатор публикации;

user1 – идентификатор пользователя, распространившего публикацию с идентификатором post_id на шаге level;

user2 – идентификатор подписчика пользователя user1, распространившего или не распространившего публикацию с идентификатором post_id на шаге level;

level – шаг распространения (см. рис. 1.3);

result – 1, если user2 распространил публикацию, 0 иначе;

feature1, …, featureN – значения атрибутов для тройки (post_id, user1, user2).

Для формирования обучающей выборки требуется получить эпизоды распространения нескольких публикаций, а затем рассчитать значения атрибутов для каждой тройки (post_id, user1, user2). Подробнее последовательность действий по формированию выборки представлена на рисунке 2.3.

Обучающие эпизоды распространения имеют следующую структуру:

- post_id: string – идентификатор распространяемой публикации;
- diffusion: array<array<int>> – массив, каждый i-й элемент которого содержит массив пользователей, распространивших публикацию на шаге i (см. рис.1.3);
- non_diffusion: array<int> – массив пользователей, которые не были активированы пользователями из массива diffusion.

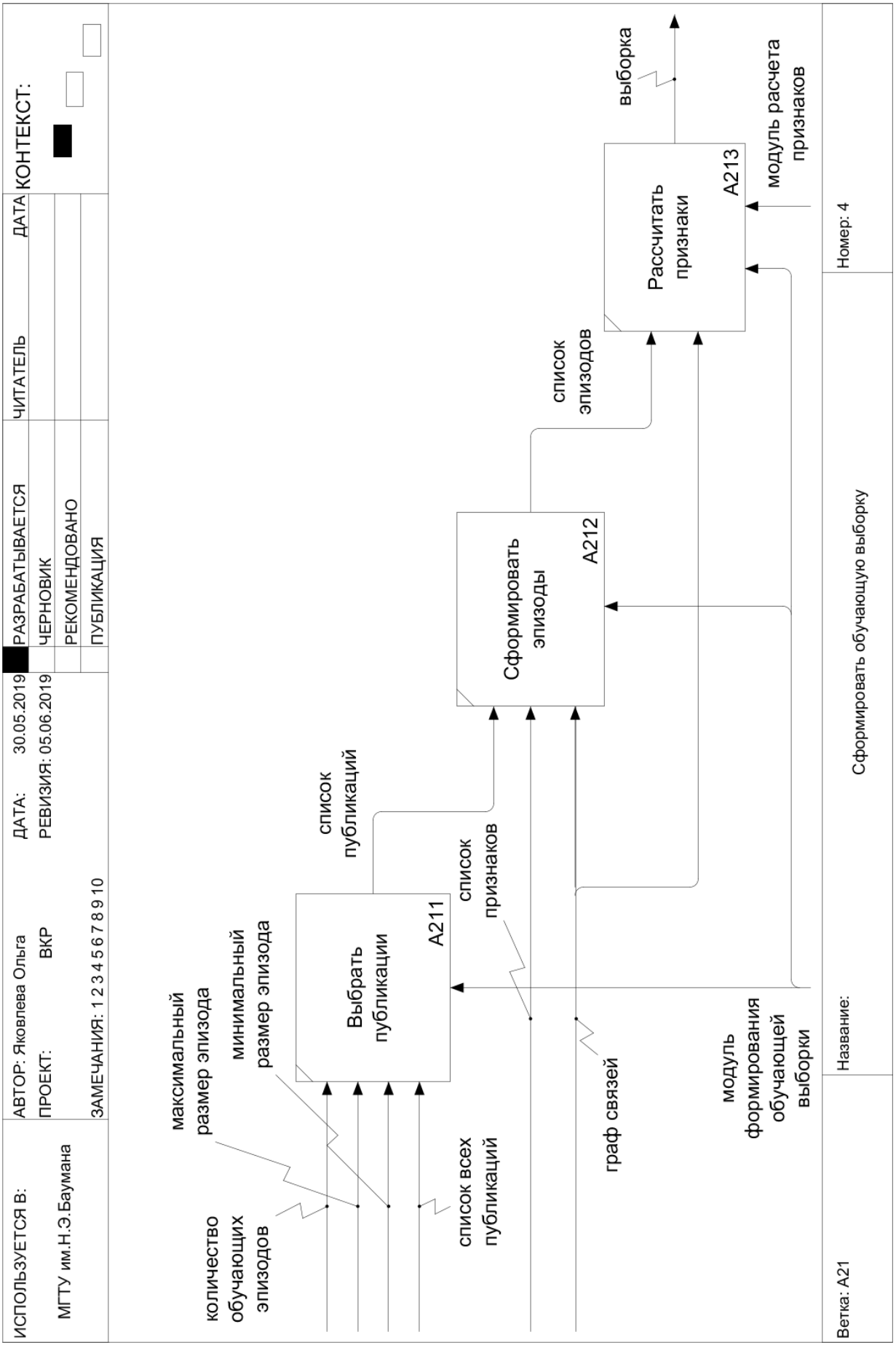

Рисунок 2.3 – IDEF0-диаграмма формирования обучающей выборки

Например, изображенному на рисунке 2.4 процессу распространения публикации с id=”1”, начинающемуся с вершины 0, будет соответствовать следующий эпизод:

```
{
    post_id = “1”;
    diffusion = [ [0], [1, 2, 3], [10, 12, 13] ];
    non_diffusion = [4, 5, 6, 7, 8, 9, 11]
}
```

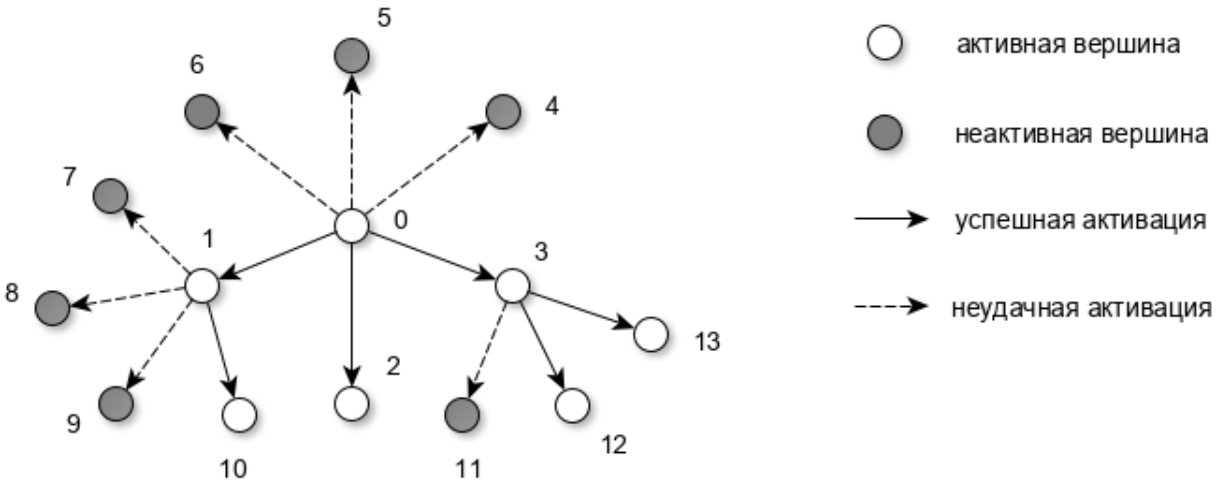

Рисунок 2.4. Процесс распространения публикации

Псевдокод получения эпизодов приведен в приложении А.

Для получения эпизода требуется восстановить каскад распространения по известной последовательности активации. Восстановление каскада будет осуществляться из предположения, что пользователя на шаге t+1 активирует тот его друг, который на шаге t последним сделал перепубликацию.

Например, для графа связей, представленного на рисунке 2.6 (а), и последовательности активации [0, 1, 2, 3, 4], восстановленный каскад распространения изображен на рисунке 2.5 (б).

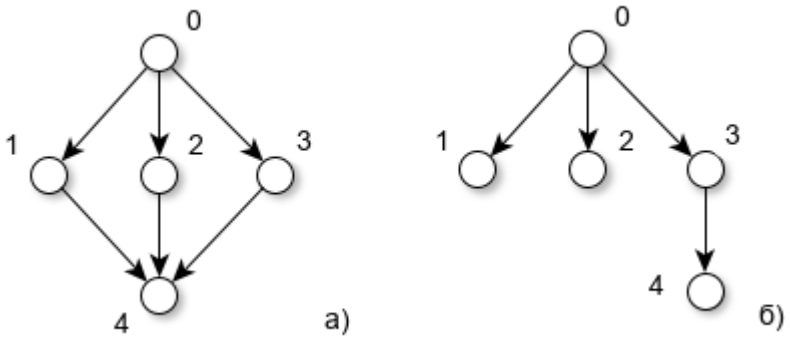

Рисунок 2.5 – Граф связей (а) и восстановленный каскад распространения (б) при последовательности активации [0, 1, 2, 3, 4]

Далее формируется матрица обучающей выборки. Последовательно обрабатываются все эпизоды. Для каждого пользователя в массиве diffusion[i] выбираются активные подписчики из массива diffusion[i+1], по формулам (12, 23-26) рассчитываются атрибуты пользователя и атрибуты подписчика, по формулам (9, 27-30) атрибуты связи и по формуле (31) атрибут публикации, в столбец level сохраняется шаг i+1, в столбец result заносится 1. Затем для каждого пользователя в массиве diffusion[i] выбираются все неактивные подписчики из массива non_diffusion, также рассчитываются атрибуты двух пользователей и связи, в столбец level заносится -1 (здесь он не играет роли), в столбец result проставляется 0.

Затем каждый столбец атрибута нормируется по максимальному элементу и алгоритмом максимизации ожидания, изложенном в 2.1, рассчитываются параметры $\boldsymbol{\theta}$.

2.2.3. Использование модели для прогнозирования

Результатом обучения модели являются коэффициенты нормирования и полученные на последней итерации алгоритма значения параметров $\boldsymbol{\theta}$. Эти данные используются для расчета вероятности распространения. Процедура построения каскада распространения итеративна и начинается с добавления в корень автора публикации. На каждой итерации перебираются все пользователи на предыдущем уровне дерева и все их подписчики. Если

вероятность распространения публикации от текущего пользователя к текущему подписчику больше 0.5, то к текущему пользователю добавляется дочерний узел (подписчик).

2.3 Структура программного обеспечения

Примерная схема программного обеспечения (ПО), реализующего описанный выше метод, представлена на рисунке 2.6.

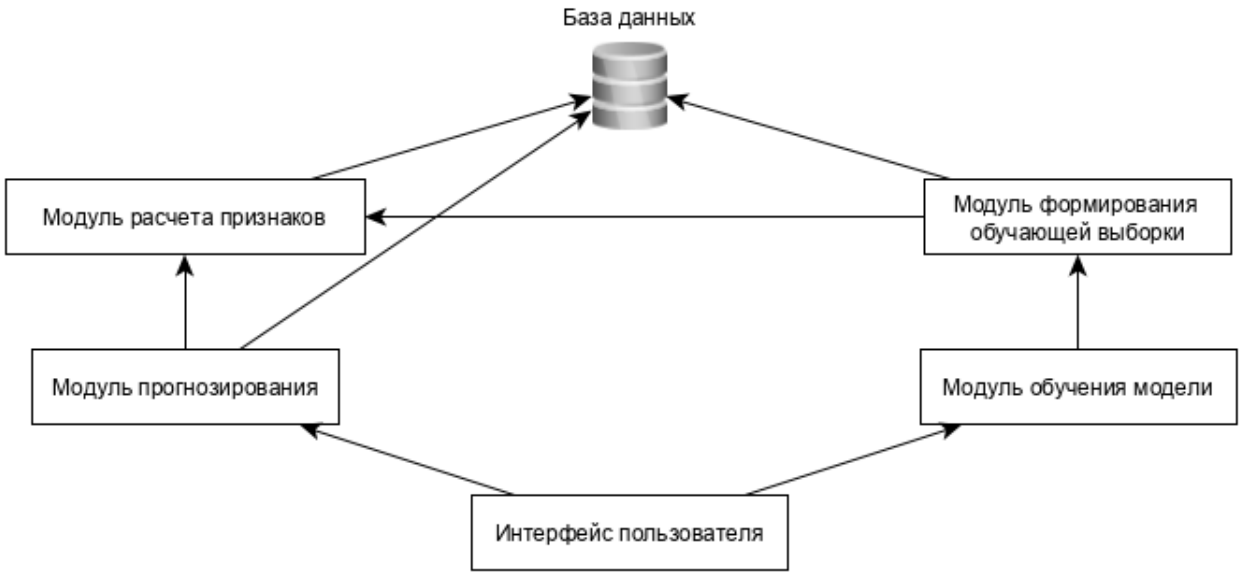

Рисунок 2.6 - Схема взаимосвязей модулей программного обеспечения

Модуль расчета признаков отвечает за расчет атрибутов пользователей, связей и публикаций.

Модуль формирования обучающей выборки отвечает за формирование матрицы данных, на которых будет обучаться модель.

Модуль обучения модели служит для расчета вектора параметров $\boldsymbol{\theta}$ с помощью алгоритма максимизации ожидания.

Модуль прогнозирования служит для построения каскадов распространения публикаций.

2.4. Вывод

В данном разделе описан разработанный метод прогнозирования. Он состоит из трех этапов: формирование списка атрибутов, значения которых влияют на вероятность распространения, настройка параметра модели

независимых каскадов, использование настроенной модели для прогнозирования. Предлагается задавать вероятность распространения как функцию трех групп атрибутов и коэффициентов, которые подбираются алгоритмом максимизации ожидания.

3 Технологический раздел

3.1 Выбор языка и среды разработки

В качестве языка программирования был выбран язык Python, в качестве среды разработки - PyCharm. Основными преимуществами языка Python, послужившими для выбора его в качестве языка реализации ПО стали:

1. Простота и скорость разработки. Разработка программных продуктов на языке Python в случае необходимости создания небольшой программы с минимальным графическим интерфейсом требует меньше временных ресурсов в сравнении с такими языками как C, C++, C#, Java.

2. Открытость. Python является открытым проектом и разрабатывается большим количеством программистов, интерпретатор языка распространяется бесплатно.

3. Кроссплатформенность. Программы, написанные на языке Python, могут быть «собраны» под большинство современных операционных систем.

4. Значительное количество открытых библиотек и модулей. В настоящее время в свободном доступе находится значительное количество сторонних библиотек для работы с файлами, базами данных, матрицами и т.д.

Выбор PyCharm в качестве среды разработки обусловлен следующими ключевыми возможностями:

- мощный и функциональный редактор кода с подсветкой синтаксиса, авто-форматированием и авто-отступами;
- помощь при написании кода, включающая в себя автодополнение, авто-импорт, шаблоны кода, проверка на совместимость версии интерпретатора языка, и многое другое;
- интеграция с системами контроля версий (VCS);
- интерактивные консоли для Python, SSH, отладчика и баз данных;
- полнофункциональный графический отладчик (Debugger);

3.2 Выбор баз данных

Для хранения рассчитанных значений атрибутов выбрана NoSQL база данных (БД) MongoDB.

MongoDB – это кросс-платформенная, документоориентированная база данных, которая обеспечивает высокую производительность и легкую масштабируемость. В основе данной БД лежит концепция коллекций и документов.

Коллекция – это группа документов MongoDB. Является эквивалентом простой таблицы в реляционной базе данных. Коллекция помещена внутри одной БД. Документ в коллекции моет иметь различные поля. Чаще всего все документы в коллекции созданы для либо одной, либо для нескольких похожих целей.

Документ – это набор пар «ключ – значение». Документ имеет динамическую схему. Это означает, что документ в одной и той же коллекции не обязан иметь один одинаковый набор полей или структуру, а общие поля в коллекции могут иметь различные типы данных.

Любая реляционная БД имеет стандартную схему, которая показывает количество таблиц и связи между ними. В MongoDB такой схемы и связи между таблицами нет.

Причины, по которым была выбрана MongoDB:

1. Количество полей, содержание и размер документов в коллекции может отличаться. Т.е. различные сущности не должны быть идентичны по структуре. Это позволит хранить значения атрибутов всех сущностей (пользователей, связей и публикаций) в одной коллекции.
2. Легкая масштабируемость. Так как в ОСС количество публикаций, пользователей и связей между ними очень велико, а разработанный метод прогнозирования распространения публикаций требует рассчитанных атрибутов для всех пользователей и их связей,

участвующих в распространении, может возникнуть необходимость в распределенном хранении данных.

3. Для хранения используемых в данный момент данных используется внутренняя память, что позволяет получать более быстрый доступ.

Требуется хранить все публикации и граф связей пользователей. Для этих целей выбрана масштабируемая полнотекстовая поисковая и аналитическая система с открытым исходным кодом Elasticsearch (ES). Она позволяет хранить большие объемы данных, проводить среди них быстрый поиск и аналитику (почти в режиме реального времени). Как правило, Elasticsearch используется в качестве базового механизма/технологии, которая обеспечивает работу приложений со сложными функциями и требованиями к поиску. ES использует в качестве поисковой основы библиотеку Lucene, которая написана на языке Java и доступна для многих платформ. Все неструктурированные данные хранятся в формате JSON, что автоматически делает ES базой данных NoSQL. Но в отличие от других баз данных NoSQL, ES предоставляет возможности поиска и многие другие функции.

Выбор Elasticsearch обусловлен следующими его достоинствами:

1. Быстрый полнотекстовый поиск. Расчет некоторых атрибутов требует полнотекстового поиска по БД и важно, чтобы этот поиск выполнялся быстро.
2. Отсутствие схемы. Данные о публикациях являются неструктурированными, поэтому БД не должна требовать наличия жестко заданной схемы.
3. Масштабируемость. Требуется хранить большой объем данных, поэтому возможность добавления “на ходу” новых серверов к уже имеющейся системе является важным преимуществом ES.

Запросы Elasticsearch выполняются с помощью Search API. Запрос и ответ представлены в виде JSON.

Запросы в Elasticsearch бывают двух типов:

1. Структурированные запросы. Используются для запроса чисел, дат, статусов и т. д. Они похожи на запросы, поддерживаемые базой данных SQL;

2. Полнотекстовые поисковые запросы. Используются для поиска в текстовых полях.

Базовая структура тела запроса:

```
{
   "size"    : // Количество документов в ответе. По умолчанию 10.
   "from"    : // Смещение результатов
   "timeout" : // Тайм-аут запроса. По умолчанию отсутствует.
   "_source" : // Поля, которые попадут в ответ.
   "query"   : { // Запрос }
   "aggs"    : { // Агрегация }
   "sort"    : { // Сортировка результата }
}
```

Основные используемые реализованными модулями полнотекстовые запросы – term, terms, bool.

Term (термин) - основной запрос в Elasticsearch. Служит для поиска документов, у которых поле FIELD точно совпадает со значением VALUE. Базовая структура запроса term:

```
{
    "query":
    {
        "term": {
            "FIELD": "VALUE"
        }
    }
}
```

Запрос terms аналогичен запросу term, за исключением того, что VALUE может принимать несколько значений. Результатом запроса являются документы, у которых поле FIELD точно соответствует одному из значений в массиве VALUES. Базовая структура запроса terms:

```
{
    "query":
    {
        "terms": {
            "FIELD": VALUES
        }
    }
}
```

Запрос bool используется для объединения разных запросов. Аналогичен использованию and/or/not для объединения разных условий в запросах к реляционным базам данных. Базовая структура запроса bool:

```
{
    "query": {
        "bool": {
            "must": [],
            "must_not": [],
            "should": []
         }
     }
 }
```

must раздел содержит все запросы *and*; *must_not* – все *not* запросы; *should* – все *or* запросы.

3.3 Основные моменты реализации

3.3.1 Используемые сторонние библиотеки

В разработке использовались следующие сторонние библиотеки языка Python:

- NumPy;
- Scipy;
- elasticsearch-py;
- PyMongo.

NumPy – библиотека, предоставляющая инструменты для эффективной работы с многомерными массивами и матрицами. Используется в модулях формирования обучающей выборки и обучения.

Scipy – библиотека для выполнения научных и инженерных расчетов. Используется в модуле обучения для решения системы уравнений методом Ньютона.

Elasticsearch-py – клиент Elasticsearch, используется в модулях формирования обучающей выборки, расчета атрибутов и прогнозирования.

PyMongo – драйвер MongoDB, используется в модулях формирования обучающей выборки, расчета атрибутов и прогнозирования.

3.3.2 Модуль формирования обучающей выборки

Модуль имеет следующие функции:

- получение списка публикаций, эпизоды распространения которых будут использованы в качестве обучающих;
- формирование обучающих эпизодов;
- формирование матрицы обучающей выборки;
- нормирование матрицы обучающей выборки.

Функция получения списка публикаций принимает 4 параметра:

- количество возвращаемых публикаций;
- максимальный размер эпизода;
- минимальный размер эпизода;
- шаг увеличения размера эпизода.

Псевдокод функции:

```
N = (max_episode_size - min_episode_size) / step
for i:=min_episode_size; i < max_episode_size; i = i + step
   size:=int(count/N)
   posts <- получить публикации, количество перепубликаций
   которых    лежит в диапазоне [i, i+step)
   добавить к результату size элементов из posts
```

Для получения публикаций, количество перепубликаций которых лежит в заданном диапазоне, выполняется запрос к Elasticsearch. В запросе все публикации группируются по идентификатору оригинальной публикации (“root_post_id”) и выбираются такие группы, размер которых не больше максимального и не меньше минимального.

Полный код функции приведен в приложении.

Функция формирования обучающих эпизодов принимает один параметр - список публикаций, и возвращает два параметра: граф связей пользователей и список эпизодов. Сначала составляются эпизоды (см. псевдокод в приложении), затем пользователи, участвующие хотя бы в одном эпизоде, добавляются в общий список. Далее полученный список используется для составления разреженной матрицы смежности:

```
Users := общий список пользователей
Len:=Length(Users)
Graph:=разреженная матрица размером Len * Len
For each user in users:
Followers:=подписчики пользователя user
For each follower in followers ∩ users:
          Graph[user, follower]:=1
```

Сохранение графа связей на этом этапе позволяет снизить дальнейшие временные издержки при обучении, поскольку запросы к Elasticsearch являются довольно затратными по времени.

3.3.3 Модуль расчета признаков

Модуль имеет набор функций для расчета каждого из признаков, перечисленных в п. 2.2.1. Общий алгоритм получения признака следующий:

```
значение признака := выполнить запрос к MongoDB
If значение признака == null:
     значение признака := рассчитать признак
     сохранить значение признака в MongoDB
return  значение признака
```

Для расчета каждого признака модуль выполняет запросы к Elasticsearch. Ниже приведены примеры таких запросов.

Для расчета активности пользователя с заданным идентификатором нужно выполнить три запроса:

1. Запрос к индексу публикаций для выбора документов, у которых поле author_id точно соответствует идентификатору пользователя, с сортировкой по времени в порядке возрастания. В запросе нужно указать

количество возвращаемых документов 1 и поле в ответе "created_at". В результате выполнения запроса будет получено время первой публикации (в POSIX). Пример запроса:

```
{
  "_source": "created_at",
  "query": {
        "term": {
                "author_id": 100
            }
        },
        "sort": [{
                "created_at": {
                    "order": "asc"
                }
            }],
        "size": 1
}
```

2. Запрос, аналогичный запросу 1, но с сортировкой по времени в порядке убывания. В результате выполнения запроса будет получено время последней публикации (в POSIX).

3. Запрос количества всех публикаций, у которых поле author_id точно соответствует идентификатору пользователя, в индексе публикаций. Пример запроса:

```
{
  "query": {
        "term": {
                "author_id": 100
            }
        }
}
```

Список всех подписчиков пользователя с идентификатором user_id можно получить с помощью следующего запроса к индексу графа связей:

```
{
    "query": {
        "term": {
            "followees": user_id
        }
    }
}
```

Стоит отметить, что Elasticsearch ограничивает максимальное количество документов в ответе (10000 по умолчанию). Поэтому для получения полного списка подписчиков приведенный выше запрос нужно выполнять в цикле, используя прокручивание (scroll).

Запрос списка друзей пользователя аналогичен запросу списка подписчиков, но для сопоставления используется поле “user”.

Запрос к индексу публикаций для подсчета количества упоминаний пользователем user_id1 пользователя user_id2:

```
{
    "query": {
        "bool": {
            "must": [{
                "term": {
                    "mentions_id": user_id2
                }
            }, {
                "term": {
                    "author_id": {
                        "value": user_id1
                    }
                }
            }]
        }
    }
}
```

Для расчета однородности по публикациям пользователей user1 и user2 выполняется 4 запроса к индексу публикаций:

1. Запрос всех публикаций пользователя user1, включая перепубликации:

```
{
    "query": {
        "term": {
            "author_id": {
                "value": user1
            }
        }
    }
}
```

2. Запрос всех публикаций пользователя user2, включая перепубликации.
3. Запрос всех перепубликаций пользователя user1:

```
{
    "query": {
        "bool": {
            "must": [{
                "term": {
                    "type": "repost"
                }
            },{
                "term": {
                    "author_id":user_id
                }
            }]
        }

    }
```

4. Запрос всех перепубликаций пользователя user2.

В таблице 3.1 приведено среднее время расчета каждого атрибута.

Таблица 3.1 – Время расчета атрибутов

Атрибут	Время расчета (с)
Активность	1.57
Количество подписчиков	0.83
Количество друзей	0.69
Количество упоминаний	0.58
Частота публикаций с упоминаниями	0.57
Однородность упоминаний	1.08
Однородность публикаций	1.94
Однородность по подписчикам	1.66
Однородность по друзьям	1.27
Частота упоминаний одного пользователя в публикациях другого	0.69
Частота присутствия хотя бы одного ключевого слова распространяемой публикации в предыдущих публикациях пользователя	1.34

3.3.4 Модуль обучения

Модуль обучения реализует метод расчета векторов параметров $\boldsymbol{\theta}$, изложенный в п. 2.1. Для расчета используется нормированная матрица обучающей выборки и матрица смежности вершин.

Псевдокод расчета:

1. инициализировать $\hat{\theta}$ начальными значениями
2. **для каждой** строки row матрицы выборки рассчитать оценку $\frac{\hat{p}_{v,w,s}}{\hat{p}_w^{(s)}}$:

```
    v := row[2]
    w := row[3]
    s := row[1]
    lvl := row[4]
    если row[5] = 0:
            p̂(v,w,s) / p̂(w)^(s) ← 1
    иначе:
            x_{v,w,m} := row[6:] # элементы с шестого и дальше
            рассчитать p̂_{v,w,s} по (15)
            Ds_rows := выбрать из матрицы выборки строки, для которых
row[4] = lvl и row[1] = s
            D_s(t) := Ds_rows [:, 1]  # взять первый элемент у всех строк
            для каждой строки g матрицы смежности:
            если g[w] != 0:
                    B(w) ← g
            рассчитать P̂_w^(s) по (19)
```

3. Найти новые значения $\boldsymbol{\theta}$, решив систему(22) методом Ньютона
4. $\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}$
5. Повторять 2-4 до сходимости $\boldsymbol{\theta}$ и $\widehat{\boldsymbol{\theta}}$.

На языке Python шаг 3 (решение системы) может быть реализован следующим образом:

```
# fp – функция, реализующая (15)
# all_data – матрица выборки
# graph – матрица смежности
# prev -  массив оценок p̂(v,w,s) / p̂(w)^(s), полученный на шаге 2 для каждой строки
матрицы выборки

def func2(teta, prev):
    global all_data # матрица выборки
    out = []
```

```
    for k in range(len(teta)):
        sump = 0
        sumn = 0
        for i, row in enumerate(all_data):
            features = row[5:].astype(float)
            if int(row[4]) == 1:
                sump = sump + (prev[i] - fp(teta,features)) * features[k]
            elif int(row[4]) == 0:
                sumn = sumn + fp(teta, features) * features[k]

        out.append(sump-sumn)

    return out

new_teta = scipy.optimize.fsolve(func2, teta, (graph, prev))
```

3.4 Сборка и запуск

Системные требования:

- Операционная система Windows 7/8/10, Linux или Mac OS X;
- Elasticsearch версии не ниже 6.3.0;
- MongoDB версии не ниже 4.0;
- Python 3.4.

Индекс графа связей в Elasticsearch должен соответствовать следующему отображению (mapping):

```
{
  "sinaweibo-graph" : {
    "mappings" : {
      "properties" : {
        "followees" : {
          "type" : "long"
        },
        "user" : {
          "type" : "long"
        }
      }
    }
  }
}
```

Индекс публикаций в Elasticsearch должен соответствовать следующему отображению (mapping):

```
{
  "sinaweibo-posts" : {
    "mappings" : {
      "properties" : {
        "author_id" : {
          "type" : "long"
        },
        "created_at" : {
          "type" : "long"
         },
        "mentions_id" : {
          "type" : "long"
        },
        "root_post_id" : {
          "type" : "text"
        },
          "text_hash" : {
          "type" : "text"
        },
        "type" : {
          "type" : "text
        }
       }
    }
  }
}
```

Для запуска необходимо выполнить следующие действия:

4. Установить необходимые библиотеки языка Python командой *pip3 install –r requirements.txt*, содержимое файла *requirements.txt* приведено ниже.

```
certifi==2019.3.9
elasticsearch==7.0.2
numpy==1.16.4
pymongo==3.8.0
scipy==1.3.0
sshtunnel==0.1.4
```

5. Отредактировать файл config.py, указав нужные настройки. Пример файла config.py приведен ниже.

```
elastic = dict(
    username="elastic",                 # имя пользователя Elasticsearch
    password="####################",    # пароль
    # хост:порт Elasticsearch без https
    url="68390afd348d4ff585878c198985b733.us-east-1.aws.found.io:9243",
    timeout=100      # таймаут запросов
```

```
)

elastic_indices = dict(
    posts_index = "sinaweibo-posts", # название индекса публикаций
    graph_index = "sinaweibo-graph"  # название индекса графа связей
)

mongo = dict(
    host="82.146.49.78",        # хост сервера с установленной MongoDB
    port=27017,                 # порт MongoDB
    ssh_username="olga",        # имя пользователя для подключения по ssh
    ssh_password="#########"    # пароль для подключения по ssh
)

mongo_database = dict(
    name = "sinaweibo"             # имя БД для хранения данных
)
```

6. Запустить модуль обучения командой *python3 em_trait.py*.

7. Запустить модуль прогнозирования командой *python3 icm.py*.

Формат входного файла модуля обучения (train_input.txt):

```
episodes_count=500
max_episode_size=1000
min_episode_size=100
step=100
```

Выходные файлы модуля обучения:

- *norm_coefs.txt* – коэффициенты нормирования;
- *teta_coefs.txt* – значения вектора параметров.

В файле *norm_coefs.txt* содержится N строк, где N – количество используемых признаков, в каждой строке записано вещественное или целое число.

В файле *teta_coefs.txt* содержится N+1 строк, где N – количество используемых признаков, в каждой строке записано вещественное число.

Входные файлы модуля прогнозирования:

- *norm_coefs.txt* – коэффициенты нормирования;
- *teta_coefs.txt* – значения вектора параметров;
- *post.txt* – идентификатор публикации, для которой будет построен каскад распространения.

Файл *post.txt* содержит одну строку, в которой записан идентификатор публикации.

Выходной файл модуля прогнозирования содержит описание графа распространения на языке DOT:

```
digraph graphname {
       id1 -> id2;
       id1 -> id3;
       id2 -> id4;
}
```

graphname совпадает с идентификатором публикации, указанным во входном файле *post.txt*.

3.5. Выводы

В данном разделе выполнено обоснование программных средств разработки и описание реализации каждого из четырех разработанных модулей. ПО реализовано на языке Python в среде разработки PyCharm, для хранения данных используются базы данных MongoDB и Elasticsearch. Приведена информация по сборке и запуску разработанного ПО.

4 Экспериментальный раздел

4.1 Критерии оценки точности разработанного метода

Для проверки точности разработанного метода предлагается оценивать следующие показатели:

1. Погрешность определения размера каскада в целом (32).

$$sc1 = \frac{abs(|D_{\text{ист}}| - |D_{\text{пред}}|)}{|D_{\text{ист}}|} * 100\% \quad (32)$$

где $|D_{\text{ист}}|$ – размер тестового каскада,

$|D_{\text{пред}}|$ – размер предсказанного каскада.

2. Погрешность определения размера каждого уровня каскада (33).

$$sc2(t) = \frac{abs(|D_{\text{ист}}(t)| - |D_{\text{пред}}(t)|)}{|D_{\text{ист}}(t)|} * 100\% \quad (33)$$

где $|D_{\text{ист}}(t)|$ – размер уровня t в тестовом каскаде,

$|D_{\text{пред}}(t)|$ – размер уровня t в предсказанном каскаде.

3. Схожесть множеств вершин в каскаде в целом (34).

$$sc3 = \frac{|D_{\text{ист}} \cap D_{\text{пред}}|}{|D_{\text{ист}} \cup D_{\text{пред}}|} \quad (34)$$

где $D_{\text{ист}}$– множество вершин в тестовом каскаде,

$D_{\text{пред}}$ – множество вершин в предсказанном каскаде.

4. Схожесть множеств вершин на каждом уровне каскада (35).

$$sc4(t) = \frac{|D_{\text{ист}}(t) \cap D_{\text{пред}}(t)|}{|D_{\text{ист}}(t) \cup D_{\text{пред}}(t)|} \quad (35)$$

где $D_{\text{ист}}$– множество вершин на уровне t тестового каскада,

$D_{\text{пред}}$ – множество вершин на уровне t предсказанного каскада.

Провести проверку точности определения вероятностей распространения затруднительно, поскольку истинные значения вероятностей неизвестны. Предлагается оценивать следующие два показателя:

- частоту “ложных срабатываний” – отношение количества неверно определенных неудачных активаций к общему количеству неудачных активаций в тестовой выборке;
- частоту “пропусков цели” – отношение количества неверно определенных удачных активаций к общему количеству удачных активаций.

При этом неудачная активация считается верно определенной, если в столбце *res* матрицы выборки (см. формат матрицы в п. 2.2.2) стоит 0, а рассчитанная вероятность распространения меньше 0.5. Аналогично, удачная активация считается верно определенной, если в столбце *res* матрицы выборки стоит 1, а рассчитанная вероятность распространения больше 0.5.

4.2 Обучающая и тестовая выборки

Для обучения модели и дальнейшей проверки метода из общего набора данных выбрано 65 каскадов распространения размерами от 50 до 600. Количество уникальных пользователей, участвующих в формировании матрицы обучающей выборки составило 17454, количество связей 208204. На рисунке 4.1 приведено распределение размеров каскадов в полученной выборке.

Рисунок 4.1 – Распределение размеров каскадов в выборке

Полученная матрица выборки состоит из 33200 строк.

Для проверки точности метода случайным образом выбрано 15 каскадов, остальные 50 будут использованы для обучения модели.

Проведено исследование влияния соотношения количества неудачных и успешных активаций в выборке на частоты “ложных срабатываний” и “пропусков цели”. Результат исследования приведен на рисунке 4.2.

Таким образом, с увеличением количества неудачных активаций относительно количества успешных снижается частота “ложных срабатываний”, но увеличивается частота “пропусков цели”, и наоборот. Если частота “ложных срабатываний” превышает частоту “пропусков цели”, прогнозируемый размер каскада будет переоценен, иначе – недооценен. Следовательно, при формировании матрицы обучающей выборки следует стремится к тому, чтобы частоты были примерно одинаковы. Как показано на рисунке 4.2, это условие достигается в случае, если количество примеров успешных и неудачных активаций одинаково.

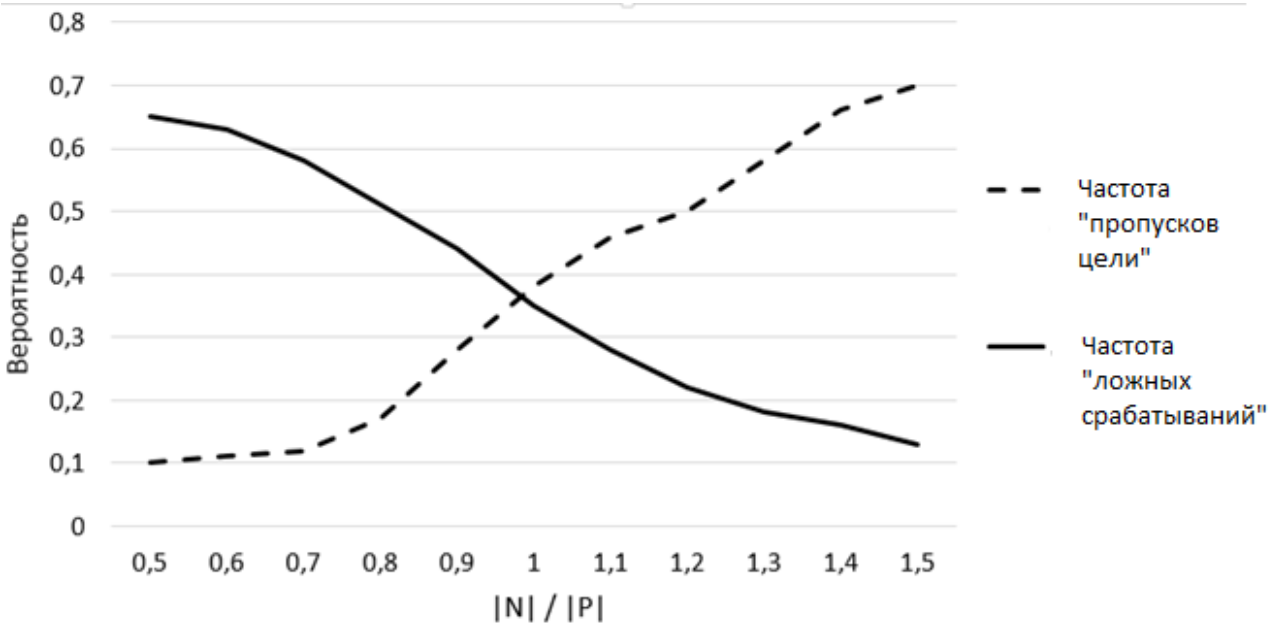

Рисунок 4.2 Зависимость частот пропусков цели и ложных срабатываний от соотношения количества активных (|P|) и неактивных (|N) пользователей

4.3 Исследование времени обучения

Проведены исследования времени расчета параметров **θ** от количества строк в матрице обучающей выборки и от количества используемых атрибутов. Результаты исследований представлены на рисунках 4.3 и 4.4 соответственно.

Рисунок 4.3 – Зависимость времени расчета параметров **θ** от количества строк в матрице выборки. Количество атрибутов зафиксировано и равно 6.

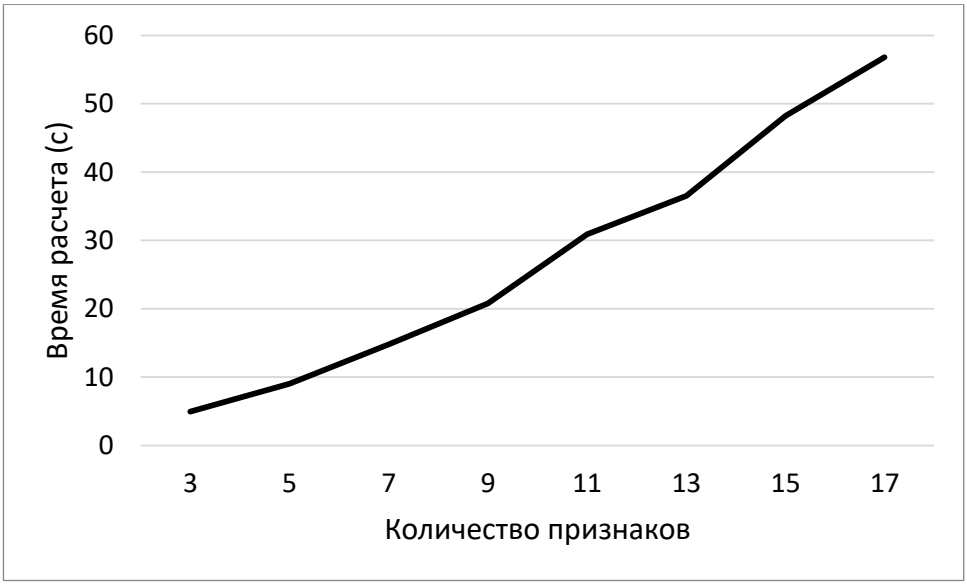

Рисунок 4.4 – Зависимость времени расчета параметров **θ** от количества используемых атрибутов. Количество строк в матрице выборки зафиксировано и равно 1000.

Таким образом, время расчета линейно зависит и от количества строк в выборке, и от количества атрибутов.

4.4 Сравнение методов определения вероятностей распространения

В таблице 4.1 приведено сравнение точности методов определения вероятностей распространения.

Обозначения:

- TP – количество верно определенных успешных активаций;
- P – общее количество успешных активаций;
- Err_P – процент “пропусков цели”;
- TN – количество верно определенных неудачных активаций;
- N – общее количество неудачных активации;
- Err_N – процент “ложных срабатываний”.

Таблица 4.1 – Сравнение алгоритмов определения вероятностей распространения

Метод	TP	P	Err_P	TN	N	Err_N
Максимизации ожидания	5496	7389	25	5496	7389	25
Логистическая регрессия	5172	7389	30	5172	7389	30
Дерево решений	5689	7389	23	5689	7389	23
Лес решений	4802	7389	45	4802	7389	45
Однослойный персептрон	4950	7389	43	4950	7389	43
Многослойный персептрон	4433	7389	40	4433	7389	40

Таким образом, точность используемого алгоритма максимизации ожидания выше, чем алгоритма логистической регрессии, используемого в [34].

4.5 Апробация метода

На рисунках 4.5 и 4.6 представлены графики зависимости истинного и предсказанного размеров каскада для двух каскадов из тестовой выборки. Анализ графиков показывает, что разработанный метод позволяет довольно точно оценивать размеры каскада на каждом шаге моделирования.

Рисунок 4.5 – Истинный и предсказанный размеры каскада №1.

Рисунок 4.6 – Истинный и предсказанный размеры каскада №2.

4.6 Выводы

В данном разделе приведены критерии оценки точности разработанного метода, описаны обучающая и тестовая выборки. Проведено исследование зависимости времени обучения модели от количества используемых атрибутов и от размера обучающей выборки. Используемый для оценки коэффициентов функции вероятности распространения алгоритм максимизации ожидания сравнен с предлагаемыми в [34] алгоритмами классификации. По результатам этого исследования можно сделать вывод, что алгоритм максимизации ожидания показывает более точные результаты, чем логистическая регрессия, используемая в методе Гиля [34]. Также исследована точность определения размера построенного каскада на каждом шаге моделирования.

Заключение

В рамках выполнения данной выпускной квалификационной работы была рассмотрена задача построения каскада распространения заданной публикации пользователя онлайновой социальной сети.

Проведена формализация предметной области, классифицированы задачи, связанные с распространением информации в социальных сетях. Сформулирована задача прогнозирования распространения публикаций и рассмотрены существующие методы её решения, использующие модель независимых каскадов.

Разработан метод построения каскадов распространения с использованием модели независимых каскадов. Он состоит из трех этапов: формирования списка атрибутов пользователей, пары связанных пользователей и пары пользователь-публикация; настройки параметра модели независимых каскадов - вероятности распространения - как функции вектора значений введенных атрибутов; моделирование.

Программное обеспечение реализовано на языке Python в среде разработки PyCharm и состоит из четырех модулей: модуля обучения, модуля расчета значений атрибутов модуля формирования матрицы обучающей выборки и модуля моделирования.

Проведено исследование точности определения размера построенного каскада распространения на каждом шаге моделирования. Его результаты показали, что метод способен определять размер каскада на каждом шаге моделирования с высокой точностью. Также проведено сравнение используемого алгоритма максимизации ожидания с алгоритмами классификации. Алгоритм максимизации ожидания показал более точные результаты.

Список использованных источников

1. Barnes J.A. Class and Committees in a Norwegian Island Parish // Human Relations, №7. 1954. P. 39-58.
2. Boyd, D. M., & Ellison, N. B. Social network sites: Definition, history, and scholarship. Journal of Computer-Mediated Communication, 13(1), article 11. 2007. P. 3.
3. Boyd D., Ellison B. Social network sites: Definition, history, and scholarship. Journal of Computer-Mediated Communication, 13(1), article 11. 2007. P. 210-230.
4. Алексеева Е.Г. Влияние через социальные сети. – М.: Фонд «ФОКУС-МЕДИА», 2010. – 200 с.
5. A.Guille, H.Hacid. Information Diffusion in Online Social Networks: A Survey. ACM SIGMOD Record, 2013.
6. Губанов Д.А., Новиков Д.А., Чхартишвили А.Г. Социальные сети: Модели информационного влияния, управления и противоборства. М.: Физматлит. - 2010. - 228 с.
7. Торопов Б.А. — Модель независимых каскадов распространения репоста в онлайновой социальной сети // Кибернетика и программирование. – 2016. – № 5. – С. 199 - 205. DOI: 10.7256/2306-4196.2016.5.20624 URL: https://nbpublish.com/library_read_article.php?id=20624
8. J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. InKDD ’09, pages 497–506, 2009.
9. D. A. Shamma, L. Kennedy, and E. F. Churchill. Peaks and persistence: modeling the shape of microblog conversations. In CSCW ’11, pages 355–358, (short paper) 2011.
10. L. AlSumait, D. Barbar´a, and C. Domeniconi. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. InICDM ’08, pages 3–12, 2008.

11. T. Takahashi, R. Tomioka, and K. Yamanishi. Discovering emerging topics in social streams via link anomaly detection. InICDM ’11, pages 1230–1235, 2011.
12. M. Gomez Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In KDD ’10, pages 1019–1028, 2010.
13. M. Gomez Rodriguez, D. Balduzzi, and B. Scholkopf. Uncovering the temporal dynamics of diffusion networks. InICML ’11, pages 561–568, 2011.
14. I. CVX Research. CVX: Matlab software for disciplined convex programming, version 2.0 beta.http://cvxr.com/cvx, sep 2012.
15. M. Gomez Rodriguez, J. Leskovec, and B. Schokopf. Structure and dynamics of information pathways in online media. In WSDM ’13, pages 23–32, 2013.
16. Kermack W.O., McKendrick A.G. A Contribution to the Mathematical Theory of Epidemics. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences. 1927, No. 115 (772), 700 p. DOI:10.1098/ rspa.1927.0118. JSTOR 94815
17. Daley DJ, Kendall DG, Stochastic rumors, J. Inst. Math. Appl. 142(1965), pp. 42-55.
18. Ажмухамедов И.М. Моделирование процесса распространения информации в социальной сети / И.М. Ажмухамедов, Ш.Ш. Иксанов // Математические методы в технике и технологиях - ММТТ. - 2014. - № 9(68). - С. 153-155.
19. J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. Marketing Letters, 2001.
20. M. Granovetter. Threshold models of collective behavior. American journal of sociology, pages 1420–1443, 1978.
21. Kempe D., Kleinberg J., Tardos E. Maximizing the Spread of Influence through a Social Network. // Proceedings of the 9-th ACM SIGKDD

International Conference on Knowledge Discovery and Data Mining, 2003.— P. 137–146.

22. Morris S. Contagion // The Review of Economic Studies.— 2000.— Vol. 67.— № 1.— P. 57–78.
23. Valente T. Network Models of the Diffusion of Innovations.— Cresskill, NJ: Hampton Press, 1995.
24. K. Saito, K. Ohara, M. Kimura, and H. Motoda. Learning Asynchronous-Time Information Diffusion Models and Its Application to Behavioral Data Analysis over Social Networks. Journal of Computer Engineering and Informatics, 2013. -P. 30-57
25. M. Cataldi, L. Di Caro, and C. Schifanella. Emerging topic detection on Twitter based on temporal and social terms evaluation. In MDMKDD ’10, pages 4–13, 2010.
26. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. InWWW ’98, pages 161–172, 1998.
27. A. Pal and S. Counts. Identifying topical authorities in microblogs. InWSDM ’11, pages 45–54, 2011.
28. R. Isea and R. Mayo-García. Mathematical analysis of the spreading of a rumor among different subgroups of spreaders. Pure and Applied Mathematics Letters (2015), Vol. 2015, pp 50–54.
29. Губанов Д. А. Обзор онлайновых систем репутации/доверия. Интернет конференция по проблемам управления. М.: ИПУ РАН, 2009. 25с.
30. De Groot M. H. Reaching a Consensus // Journal of American Statistical Assotiation. 1974. № 69. P. 118–121.
31. Робертс Ф. С. Дискретные математические модели с приложениями к социальным, биологическим и экологическим задачам. — М.: Наука, 1986.
32. Saito K., Nakano R., Kimura M. (2008) Prediction of Information Diffusion Probabilities for Independent Cascade Model. In: Lovrek I., Howlett R.J., Jain L.C. (eds) Knowledge-Based Intelligent Information and Engineering

Systems. KES 2008. Lecture Notes in Computer Science, vol 5179. Springer, Berlin, Heidelberg

33. K. Saito, K. Ohara, Y. Yamagishi, M. Kimura, and H. Motoda. Learning diffusion probability based on node attributes in social networks. In ISMIS ’11, pages 153–162, 2011.
34. A. Guille and H. Hacid. A predictive model for the temporal dynamics of information diffusion in online social networks. In WWW ’12 Companion, pages 1145–1152, 2012.
35. [Электронный ресурс] – режим доступа: https://www.web-canape.ru/business/socialnye-seti-v-2018-godu-globalnoe-issledovanie/ - дата обращения 10.05.2019
36. [Электронный ресурс] – режим доступа: https://developer.twitter.com/en/docs/basics/apps/overview - дата обращения 10.05.2019
37. [Электронный ресурс] – режим доступа https://open.weibo.com/wiki/API%E6%96%87%E6%A1%A3_V2/en - дата обращения 10.05.2019
38. [Электронный ресурс] – режим доступа https://aminer.org/influencelocality - дата обращения 10.05.2019
39. Zhang, J., Liu, B., Tang, J., Chen, T., Li, J.: Social influence locality for modeling retweeting behaviors. In: IJCAI 2013 Conference Proceedings, pp. 2761–2767

Приложение А

Псевдокод алгоритма формирования обучающих эпизодов распространения:

```
TrainEpisodes(N, RMin, RMax)
    input:
        количество обучающих эпизодов N;
        минимальный размер эпизода RMin;
        максимальный размер эпизода RMax;
        множество всех публикаций M;
        множество связей E.
    output:
        список эпизодов L.

    begin
        Posts := массив оригинальных публикаций из M

        for each Post in Posts
            Rp := массив перепубликаций публикации Post
            if Length(Rp) ≥ RMin и Length(Rp)  ≤ RMax
                then L ← MakeEpisode(Rp, p, E)
            if Length(L) ≥ N
                then break
    end


MakeEpisode(Rp, p, E)
    input:
        оригинальная публикация P;
        массив перепубликаций Rp;
        матрица связей E.
    output:
        эпизод распространения Episode.

    begin
        Episode.Post_id := P.Id
        Отсортировать Rm в хронологическом порядке
        Episode.Diffusion[0] ← P.Author_Id
        Spreaders ← P.Author_id
        for each R in Rp
            Reposter := R.Author_Id
            for i := Length(Episode.Diffusion)-1; i >= 0; i-=1
                for j := Length(i)-1; j >= 0; j-=1
                    CurrentUser := Episode.Diffusion[i][j]
                    if E[CurrentUser][Reposter] == 1
                        then
                            Episode.Diffusion[i] ← Reposter
                            Spreaders ← CurrentUser
```

```
    Len := Length(Episode.Diffusion)-1
    AllDiff := Episode.Diffusion[0] ∪ … ∪ Episode.Diffusion[Len]
    for each S in Spreaders
         AllFollowers := E[S]
         NonDiffFollowers := AllFollowers ∩ AllDiff
         for each F in NonDiffFollowers
             Episode.NonDiffusion ← F
             if Length(Episode.NonDiffusion) > Len
                 return Episode
end
```