Санкт–Петербургский государственный университет

ПИВНЕВ Игорь Алексеевич

Выпускная квалификационная работа

Использование нейросетевых методов в прикладных задачах определения мошенничества

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2015 «Прикладная математика, фундаментальная информатика и программирование»

Научный руководитель:
Кандидат физико-математических наук,
доцент кафедры теории систем управления
электрофизической аппаратурой
ГОЛОВКИНА Анна Геннадьевна

Рецензент:
Кандидат технических наук, доцент,
заведующий кафедрой технологии
программирования
БЛЕКАНОВ Иван Станиславович

Санкт-Петербург
2024 г.

Saint Petersburg State University

PIVNEV Igor Alekseevich

Graduate qualified bachelor work

Neural network methods in fraud detection applications

Education level: Bachelor's degree
Education program 01.03.02 «Applied mathematics and informatics»
Main educational program CB.5005.2015 "Applied mathematics,
fundamental informatics and programming"

Scientific supervisor:
Ph.D. in physics and mathematics,
associate professor of the department of
theory of control systems for
electrophysical equipment
GOLOVKINA Anna Gennadyevna

Reviewer:
Ph.D. in engineering, associate professor,
head of the department of programming technology
BLEKANOV Ivan Stanislavovich

Saint Petersburg
2024

Содержание

Введение 4

Постановка задачи 6

Обзор литературы 7

Глава 1. Математическая постановка задачи 11

1.1. Мошенники на сайтах-агрегаторах отзывов 11

1.2. Пространственная парадигма 12

1.3. Спектральная парадигма 15

1.4. Трансформеры 17

Глава 2. Эксперименты 19

2.1. Описание используемых наборов данных 19

2.2. Метрики 21

2.3. Эксперименты 23

2.4. Сравнение результатов 26

Выводы 28

Заключение 29

Список литературы 31

Введение

С каждым днем цифровое пространство все больше интегрируется в нашу повседневную жизнь. В современной экономике наблюдается рост влияния мошенничества, обусловленный быстрым развитием финансовых технологий, таких как необанкинг, и широким распространением социальных сетей и мессенджеров.

Согласно оценкам Центрального банка России [2], только за 2023-й год общий объем операций без согласия клиентов достиг рекордных 15.8 млрд. рублей, что на 11.48% превысило статистику предшествующего года. Более половины таких транзакций совершается с применением методов фишинга и социальной инженерии, которые активно эволюционируют в условиях развития интернета.

По данным исследования компании BI.ZONE [1], мошенничество в интернете остается одним из наиболее значимых вызовов для защиты в цифровом пространстве. За последний год число фишинговых ресурсов выросло на 86%, причем злоумышленники чаще всего создают поддельные сайты, имитирующие популярные онлайн-площадки.

Этот негативный тренд требует непрерывного совершенствования текущих и разработки новых методов выявления аномалий и подозрительных паттернов поведения участников сети. В связи с этим исследование методов обнаружения мошеннической активности становится не только актуальным, но и стратегически важным для обеспечения цифровой безопасности.

Выбор нейронных сетей обусловлен их уникальной способностью адаптироваться к сложным и динамичным структурам данных. Это делает их эффективным инструментом анализа в условиях постоянно меняющихся природы и характера цифрового мошенничества. Применение таких подходов уже оказывает существенное влияние на различные сферы, включая, но не ограничиваясь:

- Электронная коммерция: улучшение алгоритмов обнаружения мошенничества в банковских операциях и онлайн-торговле, ускорение выявления махинаций с платежами.

- Кибербезопасность: разработка инновационных методов выявления аномалий в сетевом трафике и предотвращение цифровой преступности с акцентом на оперативное реагирование на новые сценарии атак.
- Онлайн-медиа: использование нейронных сетей для нахождения ложных новостей и манипуляций с открытыми источниками данных, повышение их достоверности.

Данная работа посвящена теме определения мошеннических отзывов на сайтах-агрегаторах рецензий на товары и услуги, таких как Amazon и Yelp. Применяя решения, исследованные ниже, можно обезопасить клиентов от ошибочных покупок и укрепить доверие к платформам, что имеет важное значение для повышения общей безопасности в цифровом пространстве и защиты прав потребителей.

Постановка задачи

Целью данной выпускной квалификационной работы является анализ и сравнение различных архитектур нейронных сетей для обнаружения мошеннической активности пользователей агрегаторов отзывов о товарах и услугах.

Для достижения этой цели необходимо решить следующие задачи:

1. Провести обзор литературы и определить современные подходы на основе нейронных сетей для решения поставленной задачи.
2. Проанализировать выделенные нейросетевые архитектуры, описать их математические концепции и выявить ключевые особенности.
3. Сформулировать рассматриваемую проблему в терминах машинного обучения и определить набор метрик, которые позволят наиболее точно оценить и сравнить выбранные архитектуры.
4. Изучить доступные наборы данных для определения наиболее значимых признаков, обучить на них выбранные нейросетевые архитектуры и сравнить полученные результаты.
5. Исследовать особенности архитектур, продемонстрировавших наилучшие метрики в контексте рассматриваемой задачи.

Обзор литературы

В последние годы отмечается значительный прогресс в области использования нейронных сетей для выявления мошенничества. Особый интерес исследователей вызывают графовые архитектуры, эффективно применяемые в различных прикладных задачах. Такая структура позволяет формализовать взаимосвязи между объектами, что делает их особенно подходящими для моделирования разнообразных мошеннических схем.

Одним из ключевых элементов современных графовых нейронных сетей является так называемый модуль обмена сообщениями, известный как Graph SAmple and aggreGatE (GraphSAGE) [6]. Этот подход обобщает концепцию сверточных нейронных сетей, применяемых в обработке изображений (Computer Vision, CV), на структуры графов, что позволяет учитывать информацию о соседних узлах. В работе сперва случайным образом выбираются и затем агрегируются векторы представлений вершин из окрестности данной, что улучшает ее признаковое описание, как бы собирая информацию о всем графе. Главным недостатком такого подхода является чрезмерное усреднение скрытых представлений, также часто называемых эмбедингами (embeddings), что ухудшает разделимость классов в силу их неразличимости.

Логичным продолжением развития графовых сверток стало внедрение механизма внимания [19]. Этот модуль получил название Graph ATtention network (GAT) [20] и, будучи заимствован из области обработки естественного языка (Natural Language Processing, NLP), сразу стал неотъемлемой частью практически всех архитектур обнаружения мошенничества. Он позволил моделировать неоднородные взаимосвязи между узлами графа, акцентируя внимание на наиболее важных вершинах из окрестности данной при вычислении ее представления.

Помимо GraphSAGE и GAT, исследователи создают и менее обобщенные в своем применении модули для нейронных сетей, например, Graph Convolutional Network (GCN) [10]. Авторы данной свертки использовали графовый лапласиан для анализа процесса диффузии сигнала внутри него, показав отличное качество работы в задачах классификации вершин в графах цитирований и знаний. Однако, различные модификации этой концепции при-

меняются и в других областях, например, для получения устойчивых векторных представлений узлов и для кластеризации вершин. В дальнейшем тексте будут рассмотрены и другие актуальные подходы, часто опирающиеся или напрямую включающие в себя принципы работы перечисленных моделуй.

CAmouflage-REsistant Graph Neural Network (CARE-GNN) [5] стала первой работой, в которой рассматривалась задача обнаружения мошенников на сайтах-агрегаторах отзывов Amazon и Yelp. Авторы обратили внимание на то, что злоумышленники часто маскируют свои намерения, взаимодействуя с разными классами пользователей. Такие графы называют гетерогенными, так как в них вершины из разных групп имеют общие ребра, то есть являются инцидентными. Для борьбы с проблемой усреднения признаков исследователи предложили использовать только часть узлов из окрестности вершины, как это делается в работе GraphSAGE. Процент выбранных узлов p является параметром, который находится с помощью многорукого бандита Бернулли из области обучения с подкреплением (Reinforcement Learning, RL). Схема полученной системы представлена на рис. 1.

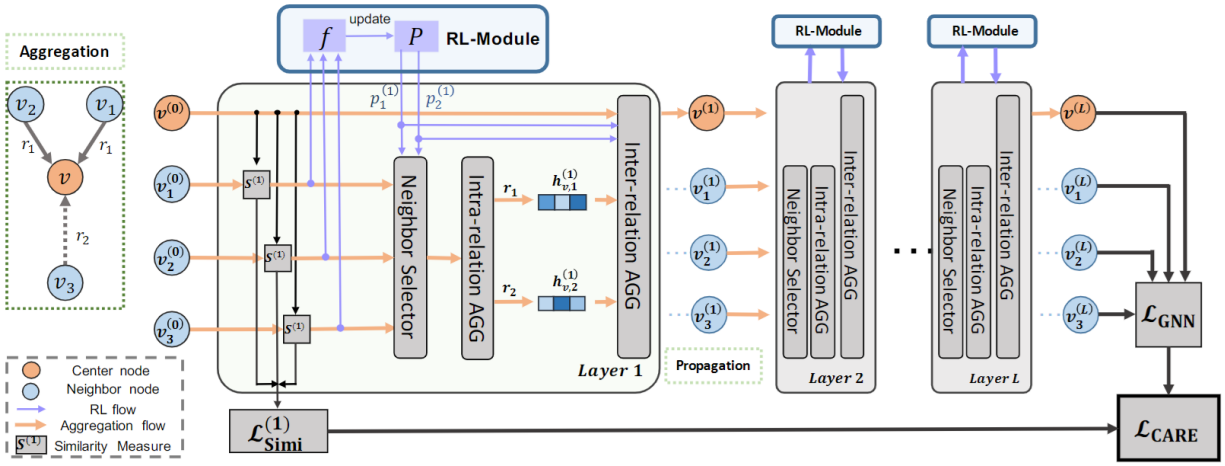

Рис. 1: Архитектура CARE-GNN: из выбранных соседей узла v только p попадают на следующий слой, что позволяет распространять информацию по графу, не слишком усредняя представления каждой вершины.

Авторы работы Neural meta-Graph Search (NGS) [14] сосредоточились на объяснимости моделей и возможности обосновать полученные предсказания. Для этого исследователи сохраняют промежуточные состояния графа после применения сверток на каждом слое и используют их для предсказания

на полученном метаграфе. Объяснимость особенно важна при использовании нейронных сетей в реальной жизни, так как невозможность разобрать полученное предсказание замедляет выявление ошибок модели.

Gated Temporal Attention Network (GTAN) [22] и Group AGgregation enhanced trAnsformer (GAGA) [21] активно используют механизм внимания GAT для получения эмбедингов вершин и ребер. В частности, авторы GAGA много внимания уделяют проблеме гетерогенных графов, предварительно группируя узлы из окрестности по классам, что приводит к более выразительным представлениям в конце. Более подробно о составляющих GAGA можно посмотреть на рис. 2.

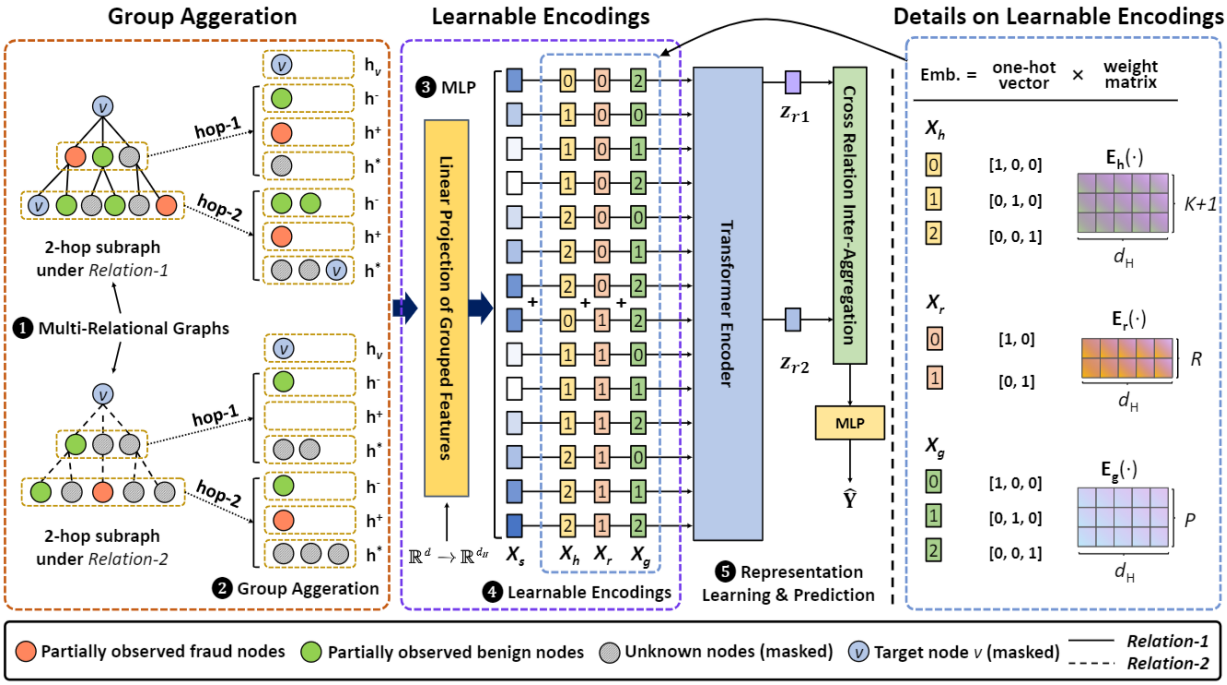

Рис. 2: Структура GAGA: под словом hop чаще всего понимают расстояние между двумя вершинами. Соседи узла относятся к hop-1, в свою очередь вершины из их окрестностей, не смежные с данной, будут считаться hop-2 и так далее.

Архитектура Dynamic Relation-Attentive Graph (DRAG) [9] использует слой внимания GAT, предварительно добавляя всем узлам петли для нахождения полного самовнимания (self-attention). Исследователи также используют подход формирования метаграфа из NGS, что улучшает производительность модели при меньшем количестве слоев и вычислений. Итоговая архитектура представлена на рис. 3.

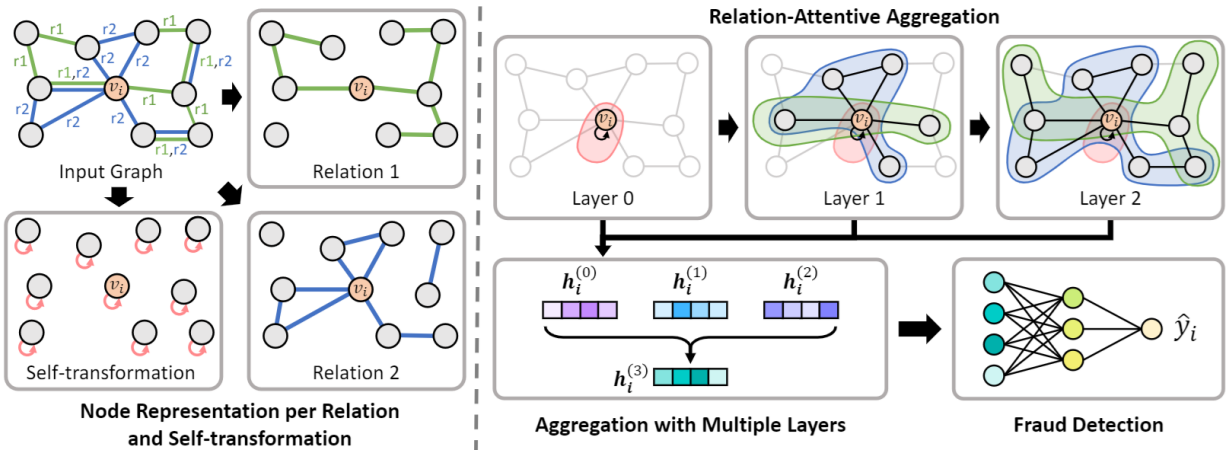

Рис. 3: Схема DRAG: разными цветами авторы изобразили несколько типов ребер. Это могут быть связи разного рода, пример такого мультиграфа будет представлен в главе 2.

Среди работ, не связанных непосредственно с обнаружением мошенничества на сайтах-агрегаторах отзывов, таких как Amazon и Yelp, можно выделить Exphormer [16], в основе которого лежит другая архитектура General Powerful Scalable graph transformer (GPS) [15]. Авторы применяют слой GAT только к части узлов, сохраняя при этом качество результата. Для этого они строят d-регулярный граф-экспандер на имеющихся вершинах, после используют несколько глобальных узлов, связанных со всеми другими, а затем добавляют полученные ребра к исходному набору. Математически авторы доказывают, что такой подход при использовании нескольких слоев приближается к GAT, но требует меньше операций для вычисления полного самовнимания. При этом полученная работа протестирована на различных задачах, включая классификацию изображений и обнаружение аномальных узлов в графе.

Глава 1. Математическая постановка задачи

В рамках данной главы сначала будет сформулирована математическая постановка рассматриваемой задачи, затем будут описаны широко применяемые парадигмы работы с графовыми данными.

1.1 Мошенники на сайтах-агрегаторах отзывов

В этой работе будем определять недобросовестных пользователей как тех, чьи рецензии (которые мы будем считать мошенническими) получают в основном низкие оценки от других пользователей. Это может свидетельствовать о попытках злонамеренно ухудшить репутацию товаров или услуг на открытых интернет-ресурсах. Будет рассмотрена широко известная задача бинарной классификации из области обучения с учителем (supervised learning), где класс 1 соответствует пользователям-мошенникам (или их отзывам, в зависимости от набора), а 0 -– всем остальным. Более подробное описание данных и метрик оценки работы моделей будет представлено в следующей главе.

Стоит отметить, что применимость рассматриваемых архитектур не ограничивается поставленной задачей и может быть успешно адаптирована и под другие темы, например, определение мошеннических банковских транзакций [3] [8] [13] и выявление аномалий в файлах записей (логах) компьютерных программ [12]. Однако такие данные обычно закрыты от публичного доступа в целях безопасности и сохранения персональной информации. Особый интерес также представляет сфера криптовалютных переводов, популярная в современном мире [4] [7], однако хоть объем данных в этой теме и значительно больше, опубликовано при этом сравнительно небольшое количество статей. Во многих перечисленных работах авторы тестировали предложенные подходы на различных задачах, что свидетельствует о широкой применимости графовых нейронных сетей в других областях [11] [17] [18] [23].

1.2 Пространственная парадигма

Пространственная парадигма работы с графовыми данными лежит в основе модуля свертки GraphSAGE [6]. Ее идея заключается в том, что у каждой вершины есть внутреннее состояние (иначе представление или эмбединг), которое можно обновить по информации из ее окрестности $N(v)$. Самой известной реализацией такого подхода, предложенной еще до распространения нейросетевых методов, стал алгоритм PageRank, который был предложен и активно использовался компанией Google для ранжирования поисковой выдачи. Для формального описания алгоритма GraphSAGE введем основные понятия, с которыми будем работать на протяжении всей главы.

В общем случае мультиграф принято представлять как набор $G = \{V, X, \{\mathcal{E}_r\}_{r=1}^{R}, Y\}$, где V – множество узлов $\{v_1, ..., v_n\}$. Каждая вершина v_i имеет d-мерный вектор признаков $x_i \in \mathbb{R}^d$, а $X = \{x_1, ..., x_n\}$ представляет собой множество всех эмбедингов узлов. $\varepsilon_{ij}^r = (v_i, v_j) \in \mathcal{E}_r$ – это ребро между v_i и v_j с отношением $r \in \{1, \ldots, R\}$. Y – это набор меток для каждой вершины в V, где $y_v \in \{0, 1\}$ и $y_v \in Y$. Набор соседей узла v обычно обозначается как $N_r(v) = \{\hat{v} \,|\, (\hat{v}, v) \in \mathcal{E}_r\}$. Граф может быть ориентированный или ненаправленный, полносвязный или многодольный, с одним отношением между узлами или множеством, а также гомо- или гетерогенный.

С учетом введенных обозначений, GraphSAGE можно записать в виде следующего алгоритма:

1. Для каждого узла v находится набор скрытых представлений соседних вершин $h_{N(v)}$, которые смежны с текущей.

2. Собранная информация агрегируется с помощью коммутативной операции $AGGR$ (например, взятие средних или максимальных значений эмбедингов) в вектор фиксированного размера.

3. Полученный вектор объединяется со скрытым представлением вершины h_v и они домножаются на матрицу W, обучаемую в процессе обратного распространения ошибки (backpropagation).

4. К результату поэлементно применяется любая нелинейная функция,

например, сигмоида:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

здесь $h_v^0 = x_v$, то есть множество всех эмбедингов узлов $X = \{x_1, ..., x_n\}$ является матрицей входных данных. Компактно алгоритм можно записать в виде следующих двух формул:

$$m_v^{t+1} = \text{AGGR}\left(\{h_w^t, w \in N(v)\}\right)$$

$$h_v^{t+1} = \sigma\left(W^{t+1}\,\text{CONCAT}(m_v^{t+1}, h_v^t)\right)$$

Полученная формула свертки использует только скрытые представления вершин и больше внимания уделяет локальному окружению, нежели глобальному положению узла во всем графе. Авторы GraphSAGE [6] показали высокое качество работы данной архитектуры в задачах, связанных с выучиванием представлений вершин, однако использование данной свертки можно встретить и в других задачах, связанных с обработкой графов, не содержащих дополнительной информации о ребрах.

В CARE-GNN [5] авторы предложили учитывать только p соседей узла, чтобы решить проблему чрезмерного усреднения. Они использовали пространственную парадигму при агрегации, но предварительно искали схожие эмбединги в скоплениях вершин с помощью следующей формулы расстояния:

$$D(l)(v, v') = \left\| \sigma\left(\text{MLP}^{(l)}\left(h_v^{(l-1)}\right)\right) - \sigma\left(\text{MLP}^{(l)}\left(h_{v'}^{(l-1)}\right)\right) \right\|$$

$$S(l)(v, v') = 1 - D(l)(v, v')$$

где $h_v^{(l-1)}$ – скрытое представление вершины v на слое $l-1$, σ – любая нелинейная функция активации, например, сигмоида, а $\text{MLP}^{(l)}$ – многослойный перцептрон на уровне l.

В NGS [14] для достижения высокой точности объединяется инфор-

мация с нескольких метаграфов, каждый из которых представляет собой промежуточное состояние исходного на всех слоях нейросети. Сначала к полученным метаграфам применяется MLP слой с нелинейной функцией активации $\text{ReLU}(x) = \max(0, x)$. Затем для каждого узла v из V у нас есть набор векторов $\{h^v_{M_1}, h^v_{M_2}, ..., h^v_{M_K}\}$, где K – параметр, заранее задающий количество метаграфов для нахождения. Мы присваиваем векторам различные веса и агрегируем следующим образом:

$$e_{M_k} = \text{MLP}\left(h^v_{M_k}\right)$$

$$\beta_{M_k} = \frac{\exp\left(e_{M_k}\right)}{\sum_{1 \le k' \le K} \exp\left(e_{M'_k}\right)}$$

$$h_v = \sum_{1 \le k \le K} \beta_{M_k} \cdot h^v_{M_k}$$

при этом h_v – скрытое представление узла, используемое для конечного прогноза, а формула β_{M_k} является многопеременной логистической функцией, которую часто называют softmax.

1.3 Спектральная парадигма

Альтернативой пространственной парадигме является спектральная, основанная на анализе процесса диффузии сигнала внутри графа с использованием матрицы смежности и лапласиана. Этот подход был представлен в работе GCN [10], авторы которой сконцентрировались на задаче классификации узлов без использования скрытых представлений ребер, которые часто являются дополнением в GraphSAGE.

Лапласиан графа — это матрица $L = D - A$, где диагональная D хранит в i-й ячейке количество исходящих из v_i ребер, а A — матрица смежности графа, где элемент a_{ij} равен числу связей $\varepsilon_{ij}^r = (v_i, v_j) \in \mathcal{E}_r$.

Лапласиан имеет неотрицательные собственные значения, среди которых количество нулевых всегда совпадает с числом компонент связности. Собственные векторы, соответствующие положительным значениям, описывают разрезы графа – его разделения пополам так, чтобы между частями было как можно меньше ребер. Этим свойством пользуются для того, чтобы проводить кластеризацию при обучении без учителя (unsupervised learning). Для этого необходимо:

1. Вычислить лапласиан L от матрицы A графа G.
2. Найти k собственных векторов, которые соответствуют наименьшим собственным значениям полученного L.
3. Из векторов сформировать матрицу размера $n \times k$, i-я строка которой будет описывать соответствующую ей вершину k параметрами.
4. Кластеризовать объекты, описываемые этой матрицей, например, с помощью алгоритма K-средних (K-means).

Алгоритм GCN очень прост с математической точки зрения и представляет собой один шаг итеративного процесса поиска собственных значений лапласиана графа: скрытые представления вершин домножаются на нормированную матрицу смежности, а также на матричный корень $D^{-\frac{1}{2}}$:

$$h^{t+1} = \theta D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}h^t$$

где h^t – это матрица размера (число вершин в графе) $\times$ (длина вектора представления), то есть к каждому каналу представлений свертка применяется отдельно, а θ является обучаемой матрицей весов. Если же мы хотим работать с несколькими каналами, то есть вместо h^t у нас имеется матрица H^t размера (число узлов) $\times$ (число каналов), а также нелинейная функция f, то формула приобретает следующий вид:

$$H^{t+1} = f\left(D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}H^t\Theta\right)$$

Таким образом, спектральный подход отлично работает в задачах, связанных с обработкой больших графов, где важно понимать относительное месторасположение вершины в нем. Однако, данная парадигма сталкивается с проблемой эффективного поиска собственных чисел матрицы смежности, а также с невозможностью учитывать скрытые представления ребер, что приводит к выбору других подходов, таких как GraphSAGE или GAT.

1.4 Трансформеры

Так называемый трансформер [19], основанный на концепции внимания (attention) и изначально спроектированный для обработки естественного языка, в последние годы повсеместно применяется и в задачах классификации графов. Объяснить это можно благодаря хорошей масштабируемости и высокому качеству работы практически с любым типом данных.

Модуль GAT [20], использующий в своей основе attention, представляет собой всего две формулы. В первой авторы вычисляют веса для внимания:

$$\alpha_{v*} = \text{softmax}\left(\text{act}\left(\mathbf{a}^T \text{CONCAT}(Wh_v^t, Wh_*^t)\right)\right)$$

здесь softmax – многопеременная логистическая функция, $\mathbf{a}^T$ – обучаемый однослойный перцептрон, CONCAT – конкатенация двух векторов и W – обучаемая матрица преобразования, общая для ключей, значений и запросов, а h_*^t является скрытым представлением вершины $v_* \in N(v)$ на слое t. act -– нелинейная функция активации, например, авторы применили:

$$\text{LeakyReLU}(x) = \begin{cases} x, & \text{если } x > 0 \\ \alpha x, & \text{если } x \leq 0 \end{cases}$$

где $\alpha \in (0, 1)$. Следующая формула используется для получения скрытых представлений узлов на новом слое с учетом полученных весов внимания:

$$h_v^{t+1} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{w \in N(v)} \alpha_{vw}^k W^k h_w^t\right)$$

при этом σ – нелинейная функция активации и K – количество голов внимания, заранее заданный параметр. Авторы используют усреднение найденных эмбедингов вершины, но в качестве альтернативы можно применить любую другую операцию агрегирования.

Множество полученных скрытых представлений узлов на последнем слое нейросети часто называют головой (Head_i) модели. В работе GTAN [22] несколько таких выходов просто объединяют, полагаясь на разную инициализацию начальных весов матрицы обучения W:

$$H = \text{CONCAT}(\text{Head}_1, \ldots, \text{Head}_{\text{att}})W_o$$

здесь CONCAT – конкатенация матриц и att – заранее заданный параметр, определяющий количество голов и размер модели.

В свою очередь, в Exphormer [16] исследователи используют скрытые представления не только вершин, но и связей между ними. Так уже приведенная формула внимания дополняется:

$$\alpha_{v*} = \text{softmax}\left(\text{act}\left(\mathbf{a}^T\text{CONCAT}(Wh_v^t, Wh_*^t, W_e e_{v*})\right)\right)$$

где e_{v*} – ребро, инцидентное вершине v и ее соседям, а W_e является дополнительной матрицей весов. Соответственно, авторы работают над уменьшением общего числа связей, по которым находится attention, так как для всего графа необходимо будет вычислить эту формулу N^2 раз, где N – количество вершин в исходном наборе.

Глава 2. Эксперименты

В этой главе сначала будет дано детальное описание наборов данных, применяемых в рассматриваемой задаче. Далее будут описаны ключевые метрики, используемые для оценки качества моделей. В заключение будут представлены эксперименты и проведено сравнение полученных результатов работы исследуемых архитектур графовых нейронных сетей.

2.1 Описание используемых наборов данных

В качестве данных для обучения и тестирования моделей используются наборы отзывов на товары и отели с ресторанами на сайтах-агрегаторах Amazon и Yelp. Впервые их применили в работе CARE-GNN [5]. Выбор обусловлен распространенностью для оценки среди исследований по определению мошенничества, а также большим объемом и реальным, а не искусственным происхождением.

В данных Amazon содержится информация о 12 тыс. людях, среди которых 1.1 тыс. (9.5%) являются мошенниками, и 4.4 млн. связях между ними. Отзывы собраны с подраздела сайта о музыкальных инструментах, причем пользователь помечался недоброжелателем, если его рецензии получали положительную оценку от других пользователей в менее чем 20% случаев. Для исследования предоставлено 25 характеристик, а пользователей объединяли связями трех типов:

1. U-P-U – оставили отзыв об одном и том же продукте.
2. U-S-V – имели одинаковый рейтинг в течение недели.
3. U-V-U – наивысшие 5% сходства текста отзывов на один и тот же товар, что измеряли с помощью алгоритма term frequency, inverse document frequency (TF-IDF).

Данные Yelp содержат 46 тыс. отзывов на рестораны и отели, среди которых 6.6 тыс. (14.5%) обозначены как мошеннические. Общее количество связей в наборе составляет 3.9 млн, представлено 32 характеристики, а рецензии объединяли по следующим признакам:

1. R-U-R – отзывы, опубликованные одним и тем же пользователем.

2. R-S-R – под одним и тем же местом, с одинаковым рейтингом.

3. R-T-R – под одним и тем же рестораном или отелем, опубликованы в одном месяце.

Таким образом, выбранные наборы данных позволяют решать задачи как определения мошеннических отзывов, так и выявления самих недоброжелателей. При этом простая структура позволяет легко представить данные в виде графа или таблицы, в зависимости от используемой модели.

Стоит отметить, что в работах часто используются и другие наборы, часто из банковской сферы. Например, Bank Account Fraud [8] представляет собой 6 млн. записей и 30 признаков, сгенерированных на основе реальных финансовых данных открытия счетов. В другом примере, Credit Card Fraud [13], авторы собрали 285 тыс. реальных транзакций и обработали их 30 характеристик с помощью метода Principal Component Analysis (PCA). Однако проблема таких данных заключается в их синтетическом (то есть искусственном) происхождении, из-за невозможности раскрыть персональную информацию. Это усложняет интерпретацию и может привести к ухудшению качества обучения модели при использовании на реальных задачах."

2.2 Метрики

Неверный выбор метрик для оценки может исказить полученные выводы, потому важно учесть все особенности данных. В случае поставленной задачи, наборы Amazon и Yelp являются достаточно несбалансированными, то есть содержат всего 9.5% и 14.5% пользователей-мошенников и их отзывов из всех данных соответственно. Учитывая это, для оценки исследуемых решений будут использоваться следующие две метрики:

1. F1-мера (f1-score):

$$\text{F1-score} = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} = \frac{\text{TP}}{\text{TP} + \frac{\text{FP}+\text{FN}}{2}}$$

   где полнота – $\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}$, точность – $\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}$ оценивают способность модели находить все примеры и точно их отмечать, а F1-мера соответсвует их среднему гармоническому. TP (true positive) и TN (true negavite) соответствуют числу верно найденных моделью 1 и 0 соответственно, а FP (false positive) и FN (false negative) – ее ошибкам.

2. Площадь по кривой (Area Under Curve, AUC): равна доле пар объектов вида (классы 1 и 0), которые модель верно упорядочила, то есть предсказанная вероятность на первом больше:

$$\text{AUC} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} \mathbb{I}[y_i < y_j] I'[f(x_i) < f(x_j)]}{\sum_{i=1}^{N}\sum_{j=1}^{N} \mathbb{I}[y_i < y_j]}$$

$$I'\left[f(x_i) < f(x_j)\right] = \begin{cases} 0, & f(x_i) > f(x_j) \\ 0.5, & f(x_i) = f(x_j) \\ 1, & f(x_i) < f(x_j) \end{cases}$$

$$\mathbb{I}\left[y_i < y_j\right] = \begin{cases} 0, & y_i \geq y_j \\ 1, & y_i < y_j \end{cases}$$

Полученную формулу AUC можно также представить как площадь под графиком, где по оси X отложена доля ложноположительных объектов, а по оси Y – доля верноположительных.

Выбранные метрики учитывают непропорциональность классов в данных, так как они усредняют разные отношения верно и неверно предсказанных объектов в каждом классе.

2.3 Эксперименты

Для выбора наилучшей архитектуры графовой нейронной сети в рамках данной работы были проведены эксперименты, в ходе которых протестированы основные расмотренные ранее подходы с различными структурными компонентами. Число выбранных работ невелико в связи с ограничениями имеющихся вычислительных мощностей.

Код реализации нейронных сетей написан на языке программирования Python с использованием таких библиотек, как PyTorch и PyG для создания основных компонентов, DGL для работы с графовыми данными, WandB для наглядного сохранения результатов экспериментов и другими. Все метрики и итоговый программный код можно найти в репозиторях, соответствующих названиям архитектур: CARE-GNN, GTAN, GAGA и DRAG, а графики – в общем WandB проекте.

Сначала наборы отзывов и данных о пользователях сайтов загружаются и преобразуются согласно требованиям соответствующего подхода. Например, в DRAG [9] ко всем узлам добавляется петля для дальнейшего нахождения полного внимания (self-attention). Затем они делятся на выборки для обучения, валидации и тестирования соответственно, после чего начинается этап тренировки, в котором используются следующие параметры:

1. CARE-GNN: 40% данных в обучающей выборке, остальные в тестовой; 30 эпох, lr = 0.01, $\lambda_1 = 2$, $\lambda_2 = 0.001$, размер пакета = 1024 и применение субдискретизации.

2. GTAN: 40% в тестовой части, 15 эпох, lr = 0.003, 128 записей в пакете, $\lambda_2 = 0.0001$, 2-хслойная нейросеть, 5 групп обучения и использование исключения.

3. GAGA: 40% данных в тренировочной выборке, 50% в тестовой и 10% в валидационной; 500 эпох, lr = 0.001, 256 и 512 – размеры пакетов в Amazon и Yelp соответственно, 2 слоя нейросети, глубина hop-4, применение исключения и 4 головы внимания.

4. DRAG: 10% записей в обучающей, 23% в валидационной и 67% в те-

стовой выборке; 1000 эпох, lr = 0.01, 1024 записей в пакете, $\lambda_2 = 0.001$ и 8 голов обучения.

где эпоха – это полный проход алгоритма по всем данным, lr – параметр, отвечающий за скорость обучения; λ_1 и λ_2 (сокращение весов, weight decay) – параметры регуляризации, размер пакета (batch size) отвечает за число записей в одном шаге обучения, субдискретизация (undersampling) применяется для исправления несбалансированности классов, группы обучения (folds) делят данные на несколько частей для большей устойчивости процесса тренировки модели, а исключение (dropout) отключает часть весов случайным выбором для лучшей обобщающей способности. Это список основных параметров, применяемых во всех работах, но есть и дополнительные, например, награды (rewards) при обучении с подкреплением в CARE-GNN [5].

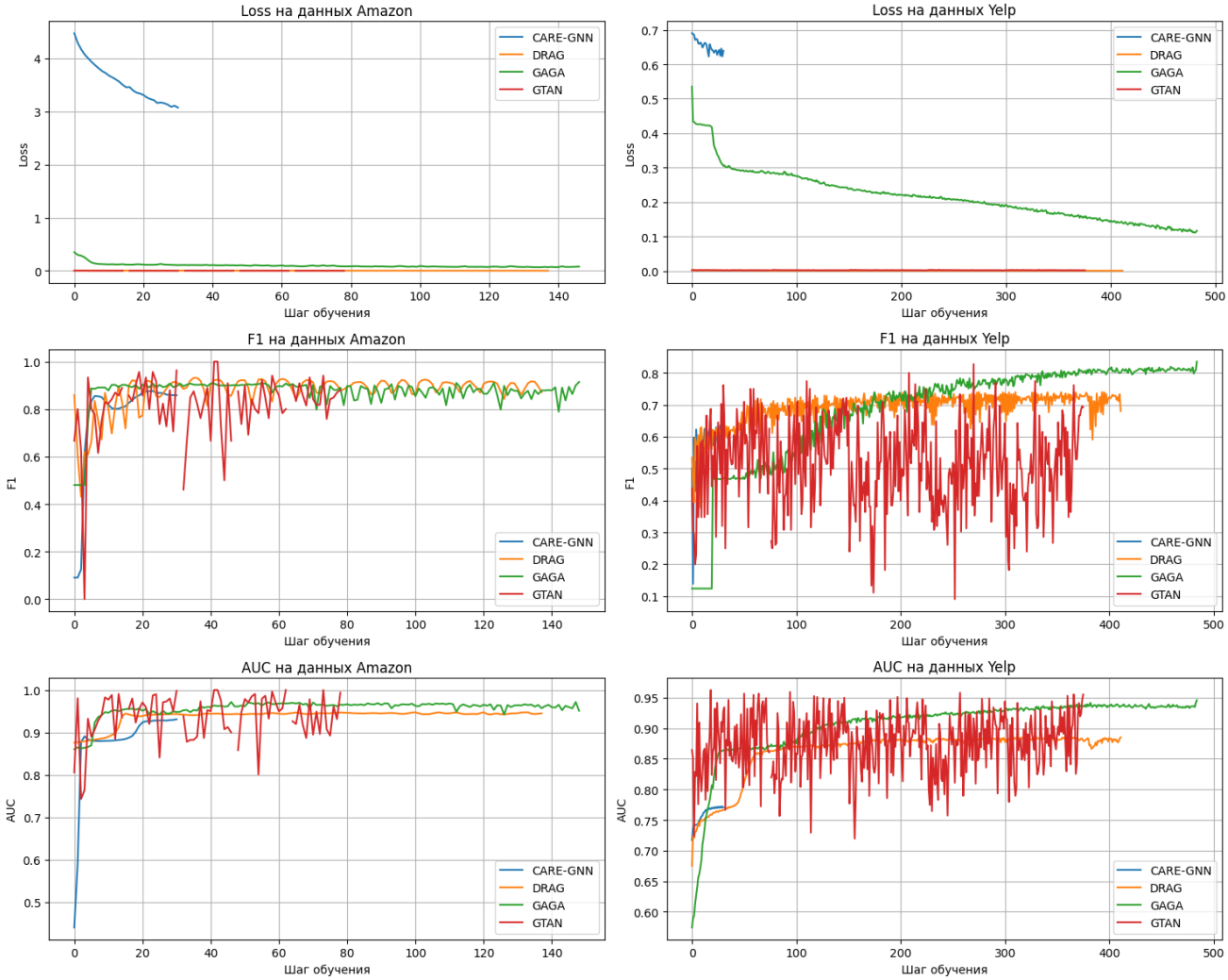

Рис. 4: Значения функции потерь (loss), f1-меры и площади под кривой на данных Amazon и Yelp соответственно.

Всего для сравнения на двух наборах было обучено 4 модели. Был проведен анализ производительности, включая сравнение времени обучения и использования ресурсов.

Из графиков на рис. 4 видно, что самая ранняя работа CARE-GNN имеет меньше всего шагов обучения и остается на уровне значения функции потерь сильно хуже других. У данной архитектуры наименьшие требования к ресурсам, что, вероятно, могло быть вызвано ограничениями на момент выхода статьи.

Модели DRAG [9] и GAGA [21] имеют больше всего шагов обучения и стальное снижение значения функции потерь при росте метрик с каждым шагом. Плато обучения достигается примерно к 350-й итерации, что можно рассматривать как момент, когда можно прекратить тренировку модели без значительной потери качества.

Графики F1-меры и AUC у модели GTAN [22] имеют стохастический вид, так как в ней применяется разбиение обучающей выборки на 5 групп, в каждой из которых один пакет содержит только 128 записей. Таким образом, обучение приобретает неоднозначный характер, когда сложно сказать, в какой именно момент тренировка закончилась и началось переобучение.

2.4 Сравнение результатов

Лучшие метрики были зафиксированы в ходе тестирования рассмотренных моделей на валидационной выборке для каждого набора данных. Все работы прошли тренировочный процесс независимо три раза, а затем были взяты усредненные значения полученных оценок и времени обучения.

Таблица 1: Полученные усредненные метрики моделей на валидации. Здесь все величины представлены в процентах для лучшей читаемости, а время обучения обозначено как число минут : секунд.

Наборы данных	**Amazon**			**Yelp**		
Модель	**F1-мера**	**AUC**	**Время**	**F1-мера**	**AUC**	**Время**
CARE-GNN	85.69	92.03	2:49	61.01	77.06	11:10
GTAN	**91.58**	**95.56**	10:28	75.57	89.24	5:47
GAGA	91.36	95.13	**0:28**	**83.34**	**94.50**	**5:31**
DRAG	89.03	94.30	5:42	68.71	87.25	16:02
NGS	92.28	97.36	–	78.28	92.18	–
GCN	65.71	81.89	–	49.63	55.04	–
GAT	53.90	74.26	–	52.28	55.19	–
GraphSAGE	83.83	91.49	–	57.81	74.09	–

Из результатов видно, что среди рассмотренных моделей GAGA с подходом разделения соседей по классам перед агрегацией показала наилучшую оценку на наборе данных Yelp при наименьшем среднем времени обучения. Выделяются также метрики упомянутой ранее работы NGS [14], однако авторы не предоставили исходный код для воспроизведения своих исследований, что затрудняет проверку результатов.

На данных Amazon лучше всего себя показала модель GTAN. Таким образом, обе работы, использующие GAT [20] в качестве ключевого модуля и с акцентом на проблеме гетерогенных графов, получили наилучшие оценки. Это указывает на значительную эффективность внимания в обработке таких данных и подтверждает значимость подхода в решении подобных задач.

В дополнение, в таблице представлены оценки оригинальных GCN [10], GAT и GraphSAGE [6] соответственно. Видно, что трансформерный и спектральный подходы в данной задаче проявили себя особенно плохо, а пространственный, наоборот, без дополнительных модификаций и доработок получает сравнительно хорошие оценки.

В целом, полученные метрики коррелируют со временем обучения и масштабами моделей, что явно позволяло им лучше справляться с рассмотренными задачами. При этом набор данных Yelp, содержащий отзывы на рестораны и отели, более сложную задачу для всех архитектур. Вероятно, наилучшим подходом для него может быть использование трансформерной модели непосредственно к тексту рецензии, возможно, даже без представления исходных данных в виде графовой структуры.

Выводы

В результате проведенного исследования были выполнены следующие задачи:

- Рассмотрена проблема и определены необходимые для оценки метрики в классических терминах области машинного обучения.
- Проанализированы актуальные подходы, применяемые к поставленной задаче и основанные на графовом представлении исходных данных.
- Оценены результаты различных архитектур на доступных и широко распространенных наборах примеров мошеннических отзывов.

Таким образом, все поставленные почти задачи были выполнены, а цель достигнута. В ходе работы не удалось только оценить и адаптировать под задачу архитектуру Exphormer [16], что было вызвано ограничениями используемых ресурсов.

Кроме того, полученные результаты демонстрируют практическую применимость рассмотренных моделей в условиях реальных данных на примере сайтов-агрегаторов рецензий Amazon и Yelp. Это подтверждает их потенциал для использования в системах модерации пользовательских отзывов. Однако исследование было ограничено доступными вычислительными ресурсами, что повлияло на количество протестированных моделей и объем данных.

Заключение

В данной работе были изучены и проанализированы актуальные подходы к определению мошеннических отзывов и самих недоброжелателей на крупнейших сайтах-агрегаторах Amazon и Yelp соответственно. Была сформулирована математическая постановка задачи, описаны главные метрики качества и воспроизведены результаты нескольких наилучших моделей на языке программирования Python. На основе проделанной работы можно сделать следующие выводы:

- Модель GAGA, разделяющая соседей вершины по их классам перед агрегацией, показала наилучшие метрики качества в обеих задачах при доступности открытого исходного кода.
- Архитектура NGS, также работающая с гетерогенными графами, продемонстрировала схожие с GAGA результаты, хотя их воспроизведение оказалось невозможным.
- Базовый подход агрегации (пространственная парадигма), представленный в GraphSAGE, показал значительное преимущество в качестве на рассматриваемых задачах по сравнению с модулями GAT и GCN.

Для дальнейшего улучшения результатов в данной области можно рассмотреть следующие направления:

1. **Оптимизация моделей**: Все исследуемые выше работы предъявляют значительные требования к ресурсам системы, на которой они запускаются. Можно изучить зависимость итогового качества от выбранных параметров обучения, чтобы добиться тех же результатов при меньших ресурсных затратах.
2. **Совмещение подходов**: К рассматриваемой графовой структуре можно добавить методы обработки естественного языка для извлечения дополнительных признаков из текстов рецензий.

3. **Масштабирование архитектур**: Улучшение основных модулей, как, например, это делают авторы Exphormer [16], может способствовать достижению больших масштабов и точности моделей.

Таким образом, результаты данной работы подтверждают перспективность использования графовых нейронных сетей для определения мошенничества и предоставляют базу для дальнейших исследований и разработок в этой области.

Рассмотренные модели могут быть интегрированы в системы модерации отзывов крупных платформ, таких как Amazon и Yelp, для автоматического выявления мошеннических рецензий и самих недоброжелателей.

Список литературы

[1] Компания BI.ZONE (2023). «Число мошеннических сайтов в 2023 году выросло на 86%». URL: https://bi.zone/news/chislo-moshennicheskikh-saytov-v-2023-godu-vyroslo-na-86/.

[2] Центральный банк России (2024). «Обзор операций, совершенных без согласия клиентов финансовых организаций». URL: https://www.cbr.ru/analytics/ib/operations_survey/2023/.

[3] Cheng D., Ye Y., Xiang S., Ma Z., Zhang Y., Jiang, C. (2023). «Anti-Money Laundering by Group-Aware Deep Graph Learning». IEEE Transactions on Knowledge and Data Engineering.

[4] Ding Z., Shi J., Li Q., Cao J. (2023). «Effective Multi-Graph Neural Networks for Illicit Account Detection on Cryptocurrency Transaction Networks». arXiv preprint, 2309.02460.

[5] Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., Yu, P. S. (2020). «Enhancing Graph Neural Network-based Fraud Detectors against Camouflaged Fraudsters». Proceedings of the 29th ACM international conference on information & knowledge management, 315–324.

[6] Hamilton W., Ying Z., Leskovec J. (2017). «Inductive Representation Learning on Large Graphs». Advances in neural information processing systems, 30.

[7] Hu S., Zhang Z., Luo B., Lu S., He B., Liu L. (2023). «BERT4ETH: A Pre-trained Transformer for Ethereum Fraud Detection». Proceedings of the ACM Web Conference 2023, 2189–2197.

[8] Jesus S., Pombal J., Alves D., Cruz A., Saleiro P., Ribeiro R. P., Gama J., Bizarro P. (2022). «Turning the Tables: Biased, Imbalanced, Dynamic Tabular Datasets for ML Evaluation». Advances in Neural Information Processing Systems, 35, 33563–33575.

[9] Kim H., Choi J., Whang J. J. (2023) «Dynamic Relation-Attentive Graph Neural Networks for Fraud Detection». 2023 IEEE International Conference on Data Mining Workshops (ICDMW), 1092–1096.

[10] Kipf T. N., Welling M. (2017). «Semi-Supervised Classification with Graph Convolutional Networks». International Conference on Learning Representations.

[11] Li S., Qiao B., Li K., Lu Q., Lin M., Zhou W. (2023). «Multi-modal Social Bot Detection: Learning Homophilic and Heterophilic Connections Adaptively». Proceedings of the 31st ACM International Conference on Multimedia, 3908–3916.

[12] Pan J., Wong S. L., Yuan Y. (2023). «RAGLog: Log Anomaly Detection using Retrieval Augmented Generation». arXiv preprint, 2311.05261.

[13] Pozzolo D. A., Caelen O., Johnson R. A., Bontempi G. (2015). «Calibrating Probability with Undersampling for Unbalanced Classification». 2015 IEEE symposium series on computational intelligence, 159–166.

[14] Qin, Z., Liu, Y., He, Q., Ao, X. (2022). «Explainable Graph-based Fraud Detection via Neural Meta-graph Search». ACM '22 International Conference on Information & Knowledge Management, 4414–4418.

[15] Rampášek L., Galkin M., Dwivedi V. P., Luu A. T., Wolf G., Beaini D. (2022). «Recipe for a General, Powerful, Scalable Graph Transformer». Advances in Neural Information Processing Systems, 35, 14501–14515.

[16] Shirzad H., Velingker A., Venkatachalam B., Sutherland D. J., Sinop A. K. (2023). «Exphormer: Sparse Transformers for Graphs». International Conference on Machine Learning, 31613–31632.

[17] Sotiropoulos K., Zhao L., Liang P. J., Akoglu L. (2023). «ADAMM: Anomaly Detection of Attributed Multi-graphs with Metadata: A Unified Neural Network Approach». 2023 IEEE International Conference on Big Data (BigData), 865–874.

[18] Tian, S., Dong, J., Li, J., Zhao, W., Xu, X., Song, B., Meng, C., Zhang, T. and Chen, L (2023). «SAD: Semi-Supervised Anomaly Detection on Dynamic Graphs». arXiv preprint, 2305.13573.

[19] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. (2017). «Attention Is All You Need». Advances in neural information processing systems, 30.

[20] Veličković P., Cucurull G., Casanova A., Romero A., Liò P., Bengio Y. (2017). «Graph Attention Networks». International Conference on Learning Representations, 1050, 20, 10–48550.

[21] Wang Y., Zhang J., Huang Z., Li W., Feng S., Ma Z., Sun Y., Yu D., Dong F., Jin J., Wang B., Luo J. (2023). «Label Information Enhanced Fraud Detection against Low Homophily in Graphs». Proceedings of the ACM Web Conference 2023, 406–416.

[22] Xiang S., Zhu M., Cheng D., Li E., Zhao R., Ouyang Y., Chen L., Zheng Y. (2023). «Semi-supervised Credit Card Fraud Detection via Attribute-Driven Graph Representation». Proceedings of the AAAI Conference on Artificial Intelligence, 37, 12, 14557–14565.

[23] Yin S., Zhu P., Wu L., Gao C., Wang Z. (2024). «GAMC: An Unsupervised Method for Fake News Detection Using Graph Autoencoder with Masking». Proceedings of the AAAI Conference on Artificial Intelligence, 38, 1, 347–355.