ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

УДК 004.85

ПОДХОД К АНАЛИЗУ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ С ПРИМЕНЕНИЕМ ГРАФОВЫХ НЕЙРОННЫХ СЕТЕЙ И КОНТРАСТИВНОГО ОБУЧЕНИЯ

Л. А. Демидова, д.т.н., профессор, профессор кафедры корпоративных информационных систем Института информационных технологий МИРЭА – Российского технологического университета, Москва, Россия;
orcid.org/0000-0003-4516-3746, e-mail: demidova.liliya@gmail.com
В. Е. Журавлев, аспирант кафедры корпоративных информационных систем Института информационных технологий МИРЭА – Российского технологического университета, Москва, Россия;
orcid.org/0009-0008-2942-0312, e-mail: vovcrane@mail.ru

Рассматривается подход к извлечению признаков из регулярных выражений с использованием графовых нейронных сетей и метода контрастивного обучения. Предложен новый подход к созданию графовых представлений регулярных выражений на основе их текстовой записи. Полученные графы отражают как содержательные, так и структурные свойства исходных регулярных выражений. Для анализа графовых представлений предлагается модель машинного обучения, основанная на графовой нейронной сети и глобальной агрегации с применением механизма внимания. Для настройки параметров модели используется метод контрастивного обучения в парадигме обучения с самоконтролем, в рамках которой осуществляется автоматическая генерация похожих графов. В ходе экспериментов используется выборка из нескольких тысяч регулярных выражений, собранных с веб-сайта Regex101. Итоговая модель, обученная на тренировочной подвыборке, оценивается с точки зрения качества и осмысленности производимых ею векторных представлений регулярных выражений. Для этого выполняется кластеризация валидационной подвыборки, результаты которой демонстрируют высокое качество извлечения признаков из регулярных выражений, подтверждая целесообразность использования графовых нейронных сетей и контрастивного обучения.

Ключевые слова: *регулярные выражения, машинное обучение, извлечение признаков, обучение представлений, графовые нейронные сети, механизм внимания, контрастивное обучение, кластеризация, k-means.*

DOI: 10.21667/1995-4565-2025-92-106-120

Введение

Регулярные выражения (РВ) представляют собой мощный инструмент для поиска и обработки текстовой информации. В настоящее время они широко используются для решения таких задач, как анализ и фильтрация текстовых данных, верификация форматов ввода, извлечение полезной информации из текстов, поиск со сложными условиями и многих других. Например, в программировании регулярные выражения активно применяются для синтаксического анализа данных [1, 2] и автоматического тестирования [3]. В сфере кибербезопасности РВ помогают в обнаружении аномалий, выявляя подозрительное поведение системы по записям в журнале ее работы [4]. Регулярные выражения также применяются в биоинформатике для поиска последовательностей ДНК, РНК, анализа белков и других аналогичных задачах [5].

Однако помимо непосредственного применения РВ, существует и ряд более высокоуровневых задач, связанных с анализом самих регулярных выражений. Например, в системах информационной безопасности могут использоваться тысячи регулярных выражений для обнаружения разных угроз, и этот список постоянно пополняется по причине появления новых видов кибератак. Для оперативного принятия решений важно обеспечить классификацию всех используемых регулярных выражений по типу выявляемых угроз, чтобы упростить анализ и восприятие результатов диагностики системы. Кроме того, имеет место и задача генерации новых РВ, похожих по своим свойствам на уже имеющиеся, чтобы можно было обнаруживать аномалии, не встречавшиеся ранее, но потенциально возможные. Все перечисленные примеры требуют качественного подхода к извлечению признаков из регулярных выражений, что само по себе является нетривиальной задачей.

Несмотря на то, что регулярные выражения представляют собой текстовые записи, они подчиняются строгому синтаксису, поэтому обладают четкой структурой. При этом стоит отметить, что именно структурные свойства в большей степени влияют на глобальный смысл всего выражения, ведь ключевые возможности РВ реализуются именно с помощью специальных синтаксических конструкций. Тем не менее наличие структурных свойств является лишь следствием соблюдения синтаксических правил, поэтому простая текстовая форма не выражает их в явном виде.

Разные логические связи внутри регулярного выражения, такие как вложенные группы, ветвления и повторения, могут быть упущены или в недостаточной степени учтены, если анализировать РВ исключительно как текст. Это ограничивает применение таких традиционных методов обработки естественного языка, как, например, мешок слов (Bag Of Words, BOW), TF-IDF, а также современные трансформерные модели, поскольку они ориентированы на анализ линейных последовательностей. Таким образом, можно заключить, что для извлечения признаков из регулярных выражений требуются альтернативные подходы, способные воспринимать не только содержание, но и структуру РВ.

Целью данной работы является исследование аспектов применения графовых нейронных сетей (Graph Neural Network, GNN) в контексте анализа регулярных выражений. Предполагается, что посредством GNN можно извлекать как содержательные, так и структурные свойства РВ, формируя тем самым выразительные векторные представления, обладающие высоким потенциалом в решении задач классификации, регрессии, кластеризации и других приложениях.

С учетом поставленной задачи в данной работе предлагается метод представления регулярных выражений в виде графов, естественным образом учитывающих их структурные свойства. Извлекать признаки из графов предлагается с помощью модели машинного обучения, основанной на графовой нейронной сети с механизмом внимания (Graph Attention network, GAT) [6] и обученной методом контрастивного обучения (Contrastive Learning, CL). Предложенный подход позволяет преобразовывать регулярные выражения в векторы фиксированного размера, отражающие как содержательную, так и структурную информацию всего РВ.

Для валидации предложенного подхода к извлечению признаков проводится кластеризация регулярных выражений, взятых из специальной выборки, не использовавшейся в процессе обучения модели. Осмысленность полученных кластеров подтверждает целесообразность использования графовых представлений для анализа регулярных выражений.

Графовые представления регулярных выражений

Как уже упоминалось, для качественного анализа регулярных выражений важно учитывать не только их текстовое представление, но и структуру, которая определяет порядок следования символов, их группировки, повторения и другие позиционные особенности. Для этой цели естественным образом подходит графовое представление РВ.

Одним из возможных графовых представлений можно назвать, например, абстрактное синтаксическое дерево (Abstract Syntax Tree, AST) [7], построение которого является алго-

ритмически разрешимой задачей для любого языка, задаваемого контекстно-свободной грамматикой (к которым, в частности, относятся языки, описываемые регулярными выражениями). Однако, прежде чем выбирать способ конструирования графов по РВ, важно рассмотреть особенности работы моделей машинного обучения, которые будут использоваться для извлечения признаков.

Среди подходов для анализа графов одним из наиболее распространенных является механизм передачи сообщений (message passing), который осуществляет распространение информации от узлов-детей к узлам-родителям. Такой подход позволяет всем узлам графа агрегировать информацию о своих соседях и благодаря этому формировать представления, учитывающие локальную иерархию связей. Механизм передачи сообщений в том или ином виде используется в большинстве современных архитектур GNN, которые различаются между собой, как правило, только особенностями выбора соседних узлов и способами агрегации полученной от них информации.

Важное свойство любой модели, использующей механизм передачи сообщений, заключается в ее невосприимчивости к порядку следования узлов-детей у каждого узла-родителя. Говоря иными словами, любая перестановка узлов, имеющих общего узла-родителя, не влияет на итоговый результат. Поэтому для анализа регулярных выражений посредством GNN необходимо использовать графовые представления, которые инвариантны к порядку следования узлов, имеющих общих соседей.

Граф AST можно понимать как дерево, в котором каждый узел представляет собой какую-либо операцию, а его дети – операнды, которые, в свою очередь, также могут быть операциями. Последовательность символов и синтаксических конструкций в РВ является неотъемлемой частью их структуры. Изменив порядок их следования, можно полностью изменить весь смысл выражения. Однако при построении AST такая последовательность преобразуется в узел-операцию конкатенации, а сами элементы последовательности – в его детей. Пример такого преобразования показан на рисунке 1.

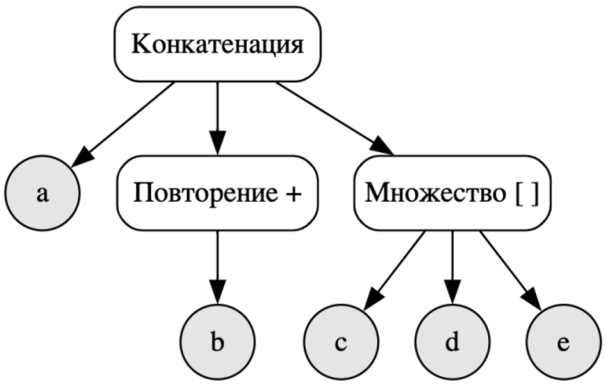

Рисунок 1 – Пример AST для регулярного выражения «ab+[cde]»
Figure 1 – Example of AST for regular expression «ab+[cde]»

Таким образом, абстрактные синтаксические деревья могут быть чувствительны к перестановкам узлов-детей, поэтому могут оказаться неудачным выбором для графового представления регулярных выражений в контексте их анализа моделью GNN.

В связи с этим в данной работе предлагается новый подход к формированию простых графовых представлений для регулярных выражений, учитывающих специфику работы GNN. Для его описания необходимо сначала рассмотреть базовое устройство РВ.

Классическая реализация регулярных выражений включает поддержку следующих четырех основных операций:

– конкатенация (например, «abc»);
– альтернация (например, «a|b|c»);
– группировка (например, «(a|b)c»);
– замыкание Клини (например, «a*»).

Любые другие операторы, такие как множества символов (например, «[abc]») или ограничения на количество повторений (например, «a{1, 3}»), могут быть представлены в виде комбинации этих четырех базовых операций. На рисунке 2 представлены правила преобразования описанных выше основных операций в графовые структуры.

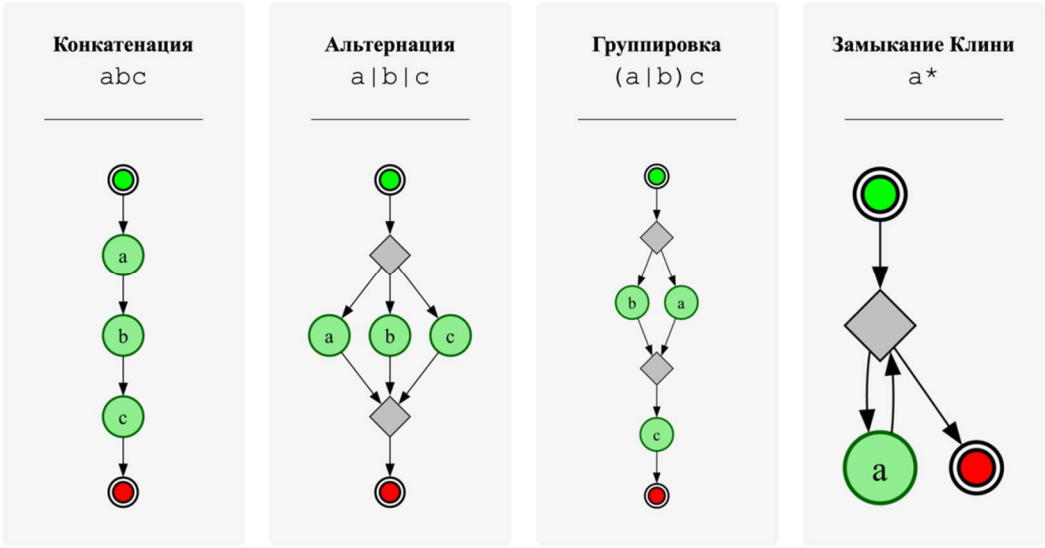

Рисунок 2 – Правила построения графовых представлений для регулярных выражений
Figure 2 – Rules for constructing graph representations for regular expressions

Получаемые с помощью предлагаемого подхода представления являются ориентированными графами (деревьями), которые естественным образом сохраняют информацию о структуре и содержании регулярных выражений.

Согласно предлагаемому подходу, дерево любого регулярного выражения обязательно содержит начальный узел, который является корнем дерева, и конечный узел, который единственный не имеет детей. Литеральные узлы, представляющие обычные символы, могут быть как положительными (означающими прямое совпадение, например «a»), так и отрицательными (означающими любой символ, кроме данного, например «[^a]»). Структурный узел является единственным узлом, для которого количество входящих и исходящих ребер может превышать единицу, поэтому он служит связующим звеном для других узлов, реализуя ветвление. Специальные узлы соответствуют таким метасимволам, как «$», «.», «\w» и другим.

На рисунке 3 представлена графовая нотация с обозначением всех перечисленных выше типов узлов, а на рисунке 4 представлен пример построенного графа для регулярного выражения «^a((bc)+|d)*e+[^f]{1,3}[a-c]$».

Рисунок 3 – Предлагаемая нотация для построения графов
Figure 3 – Proposed notation for graph construction

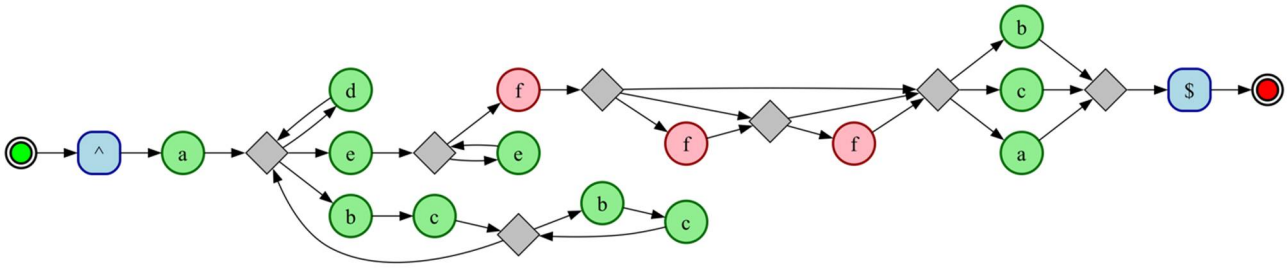

Рисунок 4 – Пример преобразования выражения «^a((bc)+|d)*e+[^f]{1,3}[a-c]$» в граф
Figure 4 – Example of converting the expression «^a((bc)+|d)*e+[^f]{1,3}[a-c]$» into a graph

Таким образом, представленный подход позволяет преобразовывать регулярные выражения в ориентированные графы, инвариантные к перестановке узлов-детей в пределах каждого узла-родителя, что делает их подходящими для графовых нейронных сетей.

Стоит отметить, что предлагаемая нотация не поддерживает такие синтаксические конструкции, как обратные связи, атомарные группировки, модификаторы, нежадные (ленивые) и сверхжадные (ревнивые) квантификаторы, а также просмотры вперед и назад. Их невозможно реализовать с помощью четырех базовых операций, и они поддерживаются не всеми диалектами регулярных выражений, поэтому не рассматриваются в данной работе [8].

Графовые нейронные сети

Графовые нейронные сети являются гибким и эффективным инструментом для анализа данных, представленных в виде графов. На сегодняшний день именно с их помощью удается достичь наилучших результатов в классификации, кластеризации, регрессии, обучении представлений и решении других классических задач как на уровне отдельных узлов и ребер, так и на уровне графов целиком [9].

Большинство современных архитектур GNN основаны на механизме передачи сообщений, уже упомянутом ранее, но отличаются друг от друга спецификой его реализации. При этом большинство популярных моделей, таких как GCN (Graph Convolutional Network) [10], GraphSAGE (Graph Sample and AGgregatE) [11] и GIN (Graph Isomorphism Network) [12], предназначены для работы с неориентированными графами, поэтому могут оказаться неэффективными для обработки регулярных выражений. Существуют разные способы реализации поддержки ориентированных графов, например раздельный анализ входящих и исходящих соседей для каждой вершины. Однако такие модификации, как правило, приводят к увеличению общей вычислительной сложности модели и не могут гарантировать осмысленного восприятия ориентированных графов, поскольку не поддерживают работу с ними естественным образом.

В связи с этим в данной работе предлагается реализовать модель для извлечения признаков из графов регулярных выражений, основанную на архитектуре GAT, которая изначально спроектирована для обработки графов с разными типами ребер, в частности, входящими и исходящими. Механизм внимания, лежащий в основе GAT, позволяет назначать разные веса для представлений соседних узлов при их агрегации, отдавая предпочтение наиболее важной для данного узла информации. Обновленное представление (вектор признаков) каждого i-го узла после применения GAT вычисляется по формуле (1):

$$\mathbf{x}_i' = \sum_{j \in \mathcal{N}_i \cup \{i\}} \alpha_{i,j} \mathbf{\Theta}_t \mathbf{x}_j, \quad (1)$$

где $\mathbf{x}_j$ – вектор признаков -го узла до применения GAT; $\mathbf{x}_i'$ – вектор признаков i-го узла после применения GAT; $\mathcal{N}_i$ – множество индексов всех узлов, исходящих из i-го; $\alpha_{i,j}$ – коэффициент внимания от i-го узла к j-му узлу, отражающий важность j-го узла для обновления состояния i-го узла; $\mathbf{\Theta}_t$ – обучаемая матрица преобразования признаков для целевых узлов.

При этом коэффициенты внимания $\alpha_{i,j}$ вычисляются по формуле (2):

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}_s^T \mathbf{\Theta}_s \mathbf{x}_i + \mathbf{a}_t^T \mathbf{\Theta}_t \mathbf{x}_j\right)\right)}{\sum_{k \in \mathcal{N}_i \cup \{i\}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}_s^T \mathbf{\Theta}_s \mathbf{x}_i + \mathbf{a}_t^T \mathbf{\Theta}_t \mathbf{x}_k\right)\right)}, \quad (2)$$

где $\mathbf{x}_i$ – вектор признаков i-го узла до применения GAT; $\mathbf{a}_s^T$ и $\mathbf{a}_t^T$ – обучаемые векторы внимания для исходных и целевых узлов соответственно, символом T в верхнем индексе показана операция транспонирования; $\mathbf{\Theta}_s$ и $\mathbf{\Theta}_t$ – обучаемые матрицы преобразования признаков для исходных и целевых узлов соответственно; $\mathcal{N}_i$ – множество индексов всех узлов, исходящих из -го; $\exp(\cdot)$ – экспоненциальная функция; $\text{LeakyReLU}(\cdot)$ – нелинейная функция активации Leaky Rectified Linear Unit [6] с коэффициентом наклона для отрицательных чисел, равным 0,2.

Поскольку применение описанных в формулах (1) и (2) преобразований позволяет для каждого узла агрегировать признаки только его непосредственных соседей, на практике обычно используют несколько слоев GAT, чтобы учитывать информацию от узлов, находящихся на большем удалении. Однако слишком большое количество таких слоев приводит к «выравниванию» признаков всех узлов из-за частой агрегации друг с другом, а также к затуханию градиентов и увеличению общей сложности модели. В связи с этим в данной статье рассматривается два слоя GAT. Первый слой состоит из четырех «голов», то есть четырех независимых GAT с разными параметрами. В результате применения этого слоя для каждого узла графа все четыре вектора признаков от каждой «головы» конкатенируются в один, а затем переходят во второй слой GAT, состоящий лишь из одной «головы». Количество «голов» было выбрано в соответствии с общепринятой практикой в области GNN и представляет собой компромисс между вычислительной сложностью модели и ее способностью к выявлению сложных зависимостей в графах регулярных выражений.

Целью данной работы является извлечение признаков из всего регулярного выражения, то есть результатом работы реализуемой модели должен быть вектор фиксированной длины. Поэтому после применения GAT необходимо агрегировать полученные для каждого узла графа признаки в единое представление. При этом важно учитывать, что регулярные выражения могут сильно отличаться друг от друга длиной и общим количеством узлов, поэтому вычисление среднего, минимального, максимального или какой-либо другой подобной статистической меры может привести к потере структурной информации. Поэтому в данной работе предлагается использовать агрегацию с вниманием (Attentional Aggregation, AA) [13], позволяющую оценивать важность тех или иных узлов при формировании общего векторного представления. Результирующее векторное представление графа $\mathbf{r}$ вычисляется по формуле (3):

$$\mathbf{r} = \sum_{n=1}^{N} \left(\frac{\exp\left(h_{\text{gate}}(\mathbf{x}_n)\right)}{\sum_{m=1}^{N} \exp\left(h_{\text{gate}}(\mathbf{x}_m)\right)} \cdot \mathbf{x}_n \right) \quad (3)$$

где N – количество узлов графа, пронумерованных от 1 до N; $\mathbf{x}_n$ – вектор признаков -го узла графа; $h_{\text{gate}}(\cdot)$ – нейронная сеть, которая вычисляет коэффициенты внимания (веса) для каждого узла на основе его признаков, то есть трансформирует вектор признаков в скалярное значение, определяющее важность данного узла; $\exp(\cdot)$ – экспоненциальная функция.

В качестве $h_{\text{gate}}(\cdot)$ в данной работе предлагается использовать простое линейное преобразование, поскольку, как будет показано далее в экспериментальном разделе, его оказывается вполне достаточно для получения качественных результатов.

На рисунке 5 представлена полная схема предлагаемой модели для извлечения признаков из регулярных выражений.

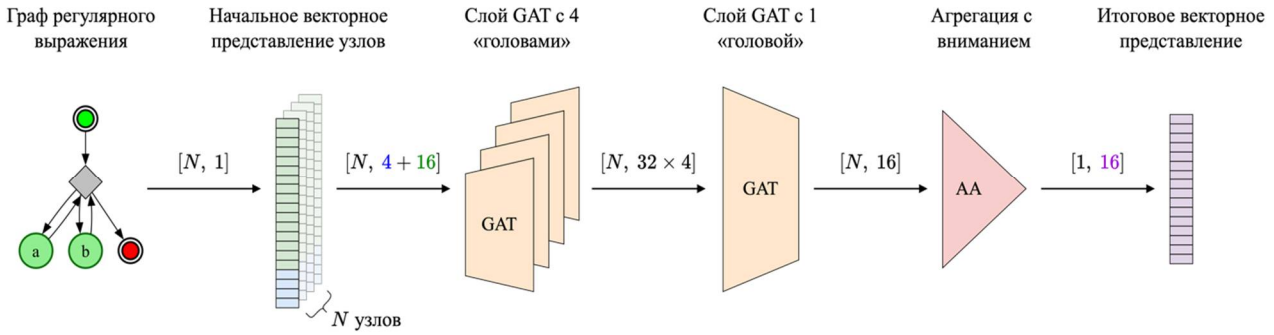

Рисунок 5 – Модель для извлечения признаков из регулярных выражений
Figure 5 – Model for extracting features from regular expressions

Для каждого преобразования на рисунке 5 в квадратных скобках указана размерность полученного результата, где N – количество узлов графа.

Начальный вектор признаков для каждого узла формируется с помощью конкатенации векторного представления типа узла с векторным представлением его значения. При этом такие представления (embeddings) являются обучаемыми параметрами. Согласно рисунку 3, в предложенной графовой нотации всего 6 типов узлов. Однако значением узла может быть абсолютно любой символ, что может привести к неадекватно большому размеру словаря векторных представлений. В связи с этим, чтобы избежать чрезмерного усложнения модели, в качестве значений рассматриваются только буквы латиницы, цифры, стандартные символы с клавиатуры, а также пустой символ для начальных, конечных и структурных узлов, особые символы для всех возможных специальных узлов и особый символ для всех остальных возможных значений, которые модель не распознает, суммарно 115 уникальных значений.

Таким образом, предложенная модель преобразует граф регулярного выражения произвольного размера в один числовой вектор из 16 элементов, отражающий как его содержание, так и структурные свойства. Размерности всех преобразований, осуществляемых в слоях модели, включая итоговый вектор признаков, были подобраны эмпирически. Как будет показано в экспериментальном разделе далее, выбранная конфигурация обеспечивает высокое качество извлечения признаков из регулярных выражений.

Контрастивное обучение

Итоговые векторные представления для графов, генерируемые моделью, должны как можно более точно отражать свойства исходных регулярных выражений. Поэтому похожие РВ должны иметь близкие представления в пространстве признаков, а непохожие – наоборот, далекие друг от друга. Эта идея лежит в основе такого метода, как контрастивное обучение (Contrastive Learning, CL).

CL представляет собой метод машинного обучения, который позволяет моделям учиться различать схожие и несхожие объекты. Контрастивное обучение широко используется в задачах компьютерного зрения, обработки естественного языка и анализа графов. В зависимости от специфики решаемой задачи применяются разные функции потерь, оптимизируемые в процессе обучения модели. Но любая из них должна удовлетворять главному принципу CL, то есть производить небольшие значения для похожих объектов (положительных примеров) и большие для непохожих (отрицательных примеров).

Однако в контексте анализа регулярных выражений неочевидно, какие объекты можно считать похожими, а какие – сильно отличающимися друг от друга. Учитывая многообразие синтаксических конструкций в РВ, изобилующее различными метасимволами и заменами, выполнение ручной разметки набора данных может оказаться затратной и неэффективной процедурой. Особенно, если имеется огромное количество регулярных выражений, состоящих из сотен символов. В связи с этим предлагается использовать парадигму обучения с самоконтролем (Self-Supervised Learning, SSL), в рамках которой графы похожих регулярных

выражений будут генерироваться автоматически с помощью случайных преобразований (аугментаций).

Нельзя не отметить, что регулярные выражения очень чувствительны к изменениям, и даже замена какого-либо одного важного символа может полностью поменять весь смысл исходного РВ. Поэтому необходимо реализовать «мягкие» аугментации, которые хоть и изменяют граф, но сохраняют его глобальные структурные свойства. В данной работе предлагаются следующие базовые аугментации для графов:

– вставка нового литерального узла со случайным значением между двумя случайно выбранными последовательно соединенными узлами, один из которых является литеральным;

– замена типа случайно выбранного литерального узла с положительного на отрицательный и наоборот;

– изменение значения случайно выбранного литерального узла на любое другое.

Предполагается, что специальные и структурные узлы наибольшим образом влияют на структуру и смысл регулярного выражения, поэтому не затрагиваются в описанных преобразованиях. Для генерации похожего графа предлагается выполнять $\max(1;\ \lceil 0{,}2N \rceil)$ случайных базовых аугментаций, где N – количество узлов в графе, а $\lceil \cdot \rceil$ означает округление в большую сторону.

Для обучения модели методом CL предлагается использовать такую функцию потерь, как InfoNCE (Information Noise-Contrastive Estimation) [14], подходящую для пакетного формата обучения, когда модель единоразово получает лишь группу объектов (пакет) из всей выборки.

Перед вычислением InfoNCE для каждого из M графов пакета последовательно генерируются похожие графы с помощью описанных ранее аугментаций, затем этот процесс повторяется еще раз. Таким образом, формируется выборка из $2M$ графов, в которой каждый -й и $(M+z)$-й графы являются аугментированными версиями одного и того же графа. Подразумевается, что каждая такая пара графов является позитивным примером, а все остальные пары – негативными примерами. Функция потерь InfoNCE вычисляется по формуле (4):

$$\mathcal{L}_M = -\sum_{z=1}^{M} \log \frac{\exp\left(\frac{\mathrm{sim}(\mathbf{e}_z;\mathbf{e}_{M+z})}{\tau}\right)}{\sum_{q=1}^{2M}\left(\exp\left(\frac{\mathrm{sim}(\mathbf{e}_z;\mathbf{e}_q)}{\tau}\right)+\exp\left(\frac{\mathrm{sim}(\mathbf{e}_{M+z};\mathbf{e}_q)}{\tau}\right)\right)} \quad (4)$$

где $\mathcal{L}_M$ – значение функции потерь; M – количество изначальных графов, а $2M$ – размер пакета после аугментаций; $\mathbf{e}_z$, $\mathbf{e}_{M+z}$, $\mathbf{e}_q$ – векторные представления, произведенные описанной ранее моделью для z-го, $(M+z)$-го и q-го графов в пакете соответственно (при этом $\mathbf{e}_z$ и $\mathbf{e}_{M+z}$ получены от аугментированных версий одного и того же графа); $\mathrm{sim}(\cdot\,;\cdot)$ – функция косинусного сходства между двумя векторами; τ – параметр температуры, используемый с целью регуляризации (в данной работе $\tau = 0{,}5$); $\exp(\cdot)$ – экспоненциальная функция; $\log(\cdot)$ – функция натурального логарифма.

В числителе формулы (4) вычисляется сходство между M позитивными парами, а в знаменателе – суммарное сходство между всеми возможными парами, включая как негативные, так и позитивные примеры. То есть, оптимизируя $\mathcal{L}_M$, модель учится различать похожие графы среди всех остальных.

Набор данных для экспериментальных исследований

Экспериментальный набор данных для обучения и валидации модели был собран с раздела «Community Patterns» веб-сайта Regex101 [15]. Для этого было проанализировано несколько сотен страниц и получено в общей сложности более 18000 регулярных выражений. Regex101 является одним из наиболее известных веб-сайтов для работы с регулярными выражениями, включая их изучение, тестирование и непосредственное применение, а раздел «Community Patterns» представляет собой обширную библиотеку РВ, постоянно пополняемую пользователями веб-сайта. Учитывая перечисленные особенности, можно считать, что

собранные данные обладают высокой степенью разнородности, что может оказаться полезным в контексте контрастивного обучения.

Однако перед использованием собранных данных необходимо выполнить их предобработку, поскольку полученные регулярные выражения могут содержать синтаксические ошибки или неподдерживаемые конструкции (обратные связи, модификаторы и другие). Кроме того, среди РВ могут встречаться дубликаты. При этом важно отметить, что у регулярных выражений могут быть разные строковые представления, но идентичные графы. Например, у «a+» и «aa*» одинаковое графовое представление, поскольку оба выражения являются полностью эквивалентными, а предложенный в данной работе подход к конструированию графов автоматически учитывает такие случаи.

В ходе анализа набора данных было установлено, что около 96 % всех регулярных выражений состоит из не более чем 1000 узлов в графовом представлении. С учетом этого факта из набора данных были удалены РВ, состоящие из 1001 и более узлов, поскольку такие графы могут сильно замедлить процесс обучения, но составляют при этом меньшинство, поэтому не вносят значительного вклада в репрезентативность набора.

В итоге после описанной выше фильтрации осталось 9815 уникальных регулярных выражений. Для валидации представленной в данной работе модели полученный набор был разделен на тренировочную и валидационную выборки в соотношении 80:20 соответственно. Валидация заключается в оценке качества и осмысленности кластеризации на валидационной выборке, которая не использовалась в процессе обучения модели.

Обучение и валидация модели

Процесс обучения и валидации представленной в данной работе модели был выполнен с использованием языка программирования Python 3.12 в среде Jupyter Lab 4.1.6. В ходе экспериментов использовался следующий компьютер: MacBook Pro 13 2017 A1708 [процессор: Intel(R) Core(TM) i5-7360U CPU 2.3 ГГц, 2.3 ГГц, 2 ядра; оперативная память: 8 Гб; 64-разрядная операционная система]. Сама модель GNN была реализована с помощью библиотек PyTorch [16] и PyTorch Geometric [17].

Модель обучалась в течение 10 эпох на тренировочной выборке с размером пакета $M = 32$. Для оптимизации функции потерь $\mathcal{L}_M$ использовался метод Adam (Adaptive Moment Estimation) [18] со значением параметра скорости обучения 0,01. На рисунке 6 показано изменение значений $\mathcal{L}_M$ в ходе эпох обучения.

В результате была получена модель, извлекающая из произвольного регулярного выражения признаки в виде вектора из 16 значений. Для проверки осмысленности производимых векторных представлений предлагается выполнить кластеризацию на валидационной выборке с помощью алгоритма k-means (k-средних) [19]. При этом важно отметить, что в качестве меры близости векторов должна использоваться косинусная метрика сходства, поскольку именно она применялась при вычислении функции потерь в процессе обучения модели.

Количество кластеров K в алгоритме k-means является гиперпараметром, то есть заранее неизвестно. Для выбора оптимального значения K можно использовать такую метрику, как индекс кластерного силуэта (silhouette score) [20], которая позволяет оценить как степень разделенности кластеров, так и компактность их внутренней структуры. Чем больше ее значение, тем более качественным считается разбиение данных. Для каждого значения K от 2 до 30 алгоритм k-means был независимо запущен 100 раз, а оптимальное количество кластеров было выбрано по максимальному значению индекса кластерного силуэта. На рисунке 7 представлены результаты произведенных запусков.

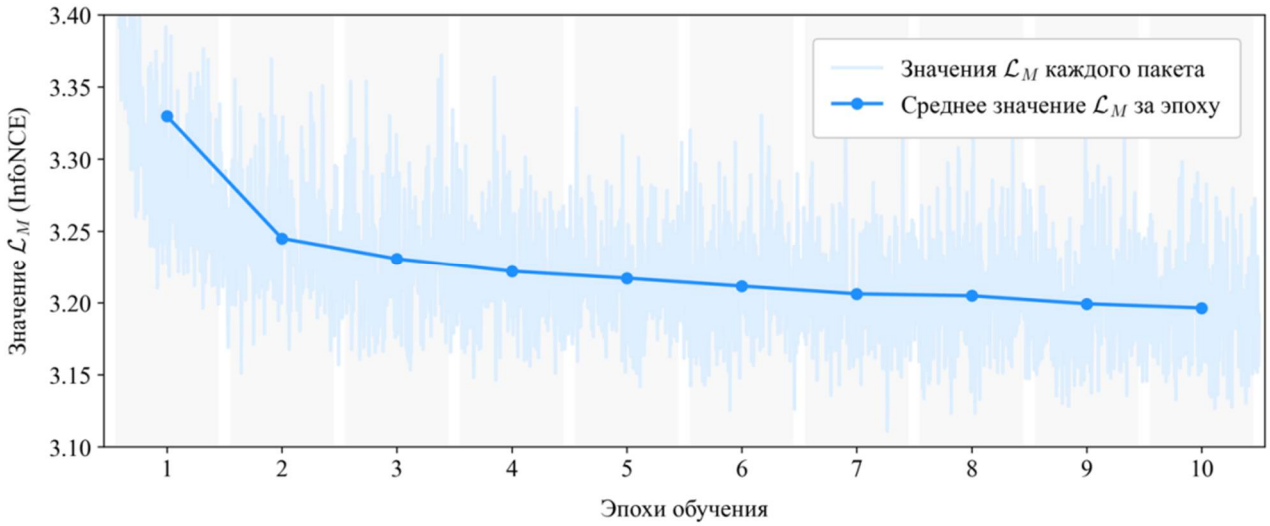

Рисунок 6 – Уменьшение значений $\mathcal{L}_M$ в ходе эпох обучения
Figure 6 – Decreasing $\mathcal{L}_M$ values during epochs of learning

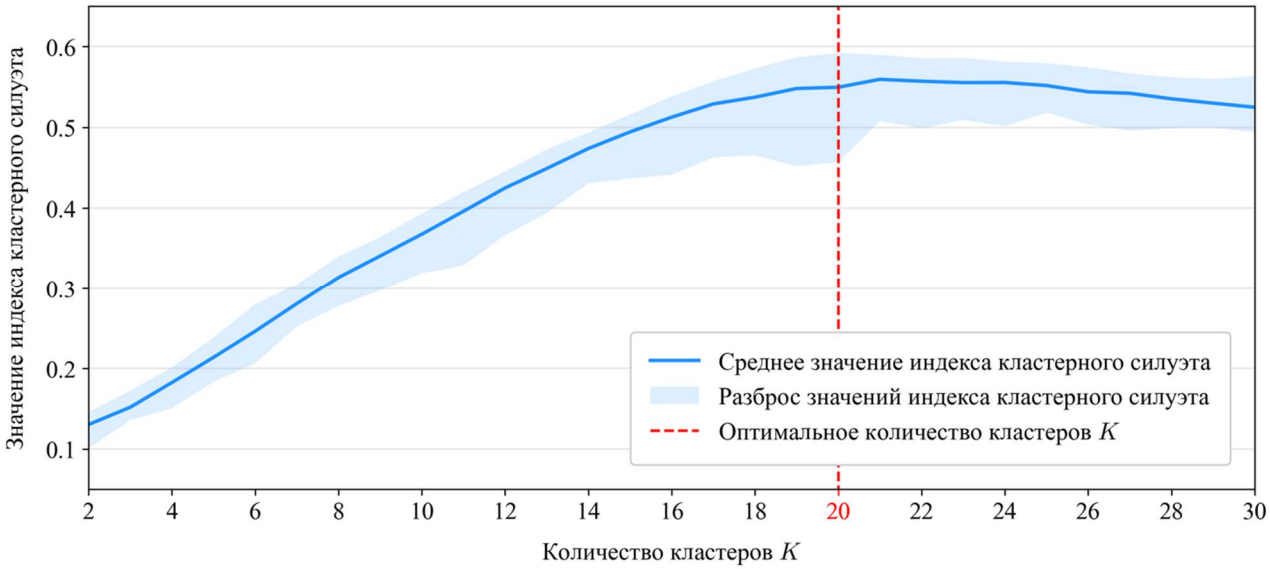

Рисунок 7 – Поиск оптимального количества кластеров
Figure 7 – Finding optimal number of clusters

В соответствии с графиком, представленном на рисунке 7, оптимальным количеством кластеров для анализируемого набора данных можно считать $K = 20$, для которого максимальное значение индекса кластерного силуэта составило 0,592. Поскольку каждое регулярное выражение преобразуется в числовой вектор из 16 элементов, кластеризация осуществляется в 16-мерном пространстве признаков, что делает невозможным визуализацию результатов в исходном виде. Поэтому для визуального представления данных предлагается использовать алгоритм UMAP (Uniform Manifold Approximation and Projection) [21], выполняющий нелинейное снижение размерности с $\mathbb{R}^{16}$ до $\mathbb{R}^2$. На рисунке 8 представлена визуализация результатов кластеризации. Для центров кластеров была построена диаграмма Вороного, разбивающая двухмерную плоскость на области, окрашенные в соответствующие кластерам цвета.

Визуально можно заметить, что некоторые регулярные выражения формируют отчетливые кластеры, находящиеся на удалении от других, что подтверждает способность моделей различать похожие и непохожие РВ.

Для оценки осмысленности сформированных кластеров предлагается рассмотреть входящие в них регулярные выражения. На рисунке 9 представлены некоторые РВ, входящие в два разных кластера.

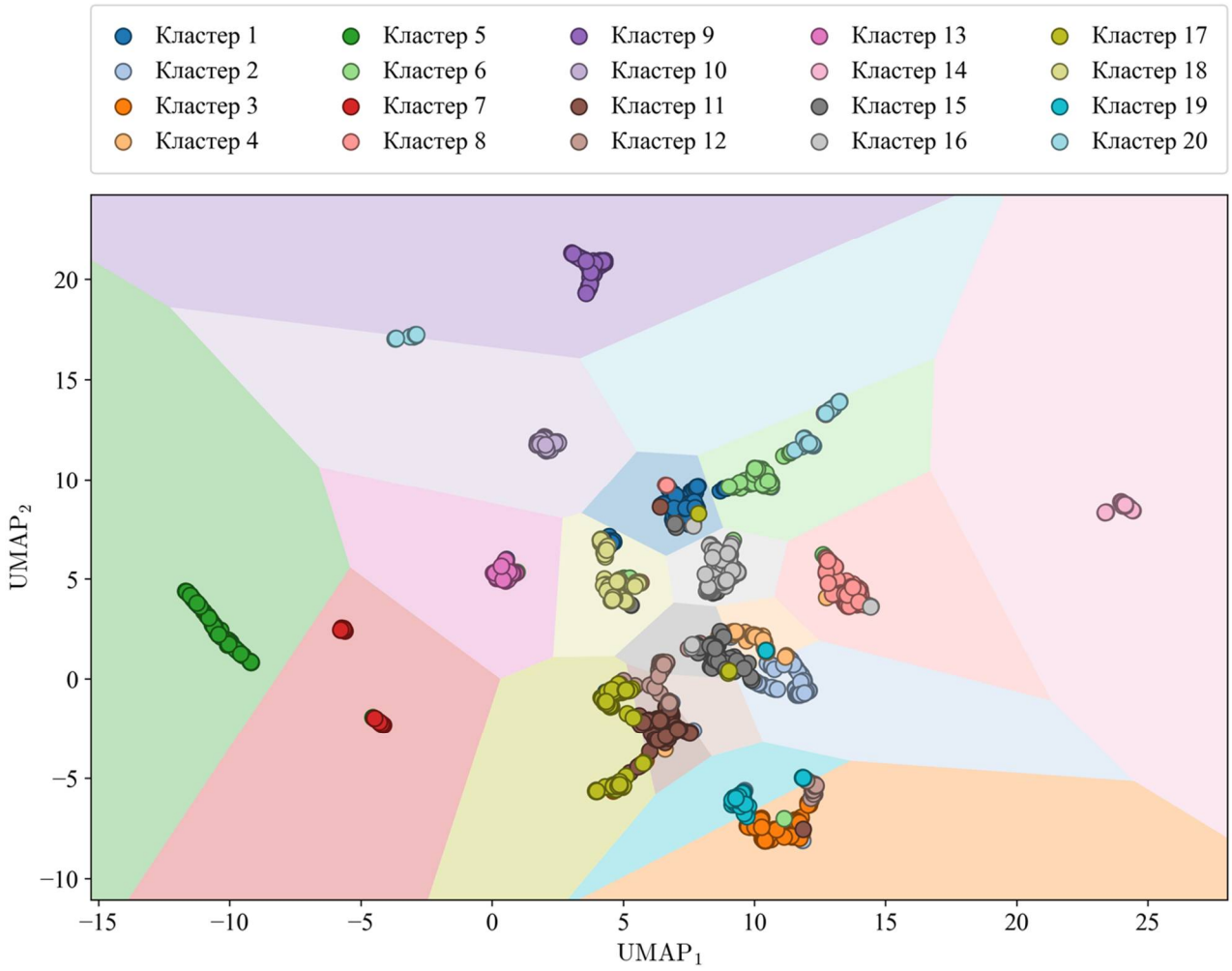

Рисунок 8 – Визуализация результатов кластеризации с помощью алгоритма UMAP
Figure 8 – Visualization of clustering results using UMAP algorithm

По рисунку 9 видно, что регулярные выражения в пределах одного кластера действительно обладают схожими чертами как в структурном, так и в содержательном смысле. Например, в четвертом кластере можно наблюдать графы, состоящие преимущественно из обычных и специальных символов, объединенные в небольшие структуры с несколькими развилками. Третий кластер, наоборот, выделяется длинными литеральными последовательностями, которые не встречались в предыдущем кластере. Остальные кластеры также обладают явно различимыми свойствами, присущими всем регулярным выражениям, входящим в них. Есть кластеры, почти полностью состоящие из коротких регулярных выражений, построенных из литеральных узлов, а есть скопления огромных графов, в которых количество развилок может превышать несколько десятков ветвей.

Таким образом, по результатам кластеризации можно сделать вывод об осмысленности получаемых с помощью модели GNN векторных представлений для регулярных выражений.

Рисунок 9 – Примеры регулярных выражений из разных кластеров
Figure 9 – Examples of regular expressions from different clusters

Заключение

В данной работе был предложен подход к преобразованию регулярных выражений в графовые представления, естественным образом отражающие не только содержание, но и структуру РВ. Такой подход обеспечивает возможность качественного извлечения признаков, релевантных для дальнейшего решения высокоуровневых задач, связанных с анализом регулярных выражений.

Для обработки графов была предложена модель, состоящая из двух слоев графовой нейронной сети с механизмом внимания (GAT) и слоя агрегации всех узлов с глобальным вниманием (AA).

Настройка параметров модели осуществлялась методом контрастивного обучения (CL), основная идея которого заключается в формировании близких векторных представлений для похожих регулярных выражений и далеких для непохожих. При этом была реализована парадигма обучения с самоконтролем (SSL), в рамках которой был предложен подход к случайным аугментациям графов, что позволило автоматически генерировать похожие графы без необходимости ручной разметки данных.

Набор данных для экспериментов был собран с веб-сайта Regex101 и представляет собой множество разнородных РВ, предназначенных для решения несвязанных друг с другом задач. Для обучения использовалась тренировочная выборка из 7852 регулярных выражений, а для валидации – специальная выборка из других 1963 РВ. В результате была получена модель, преобразующая произвольное регулярное выражение в компактное векторное представление размерностью 16.

Оценка качества модели проводилась посредством кластеризации векторов признаков регулярных выражений из валидационной выборки. Полученные результаты подтвердили, что представленная модель формирует осмысленные векторные представления, отражающие как содержательные, так и структурные характеристики регулярных выражений.

Направления дальнейших исследований могут включать в себя усовершенствование предложенной в данной работе модели, а также рассмотрение возможности использования альтернативных форматов графовых представлений регулярных выражений. В частности, перспективным подходом можно назвать объединение абстрактного синтаксического дерева

(AST) с позиционным кодированием, что может позволить модели учитывать порядок узлов-детей в пределах каждого узла-родителя. Кроме того, применение алгоритма разбиения BPE (Byte-Pair Encoding) [22] для анализа литеральных узлов может расширить множество распознаваемых моделью символов и повысить качество извлечения признаков из регулярных выражений.

Библиографический список

1. **Козлов С.В., Светлаков А.В.** Применение регулярных выражений для обработки текстовых данных // International Journal of Open Information Technologies. 2022. № 9.

2. **Панамарева О.Н., Гребенщиков Д.А., Сухарев Д.А.** Многоцелевая фильтрация текста с использованием регулярных выражений // Известия Тульского государственного университета. Технические науки. 2024. № 7. С. 284-288.

3. **Wang P., Brown C., Jennings J.A., Stolee K.T.** Demystifying regular expression bugs: A comprehensive study on regular expression bug causes, fixes, and testing // Empirical Software Engineering. 2021, № 27.

4. **Kozik R., Choraś M., Renk R., Hołubowicz W.** Modelling HTTP Requests with Regular Expressions for Detection of Cyber Attacks Targeted at Web Applications // International Joint Conference SOCO'14-CISIS'14-ICEUTE'14. 2014. С. 527-535.

5. **Yan T., Yoo D., Berardini T.Z., Mueller L.A., Weems D.C., Weng S., Cherry J.M., Rhee S.Y.** PatMatch: a program for finding patterns in peptide and nucleotide sequences // Nucleic Acids Research. 2005. № 33. С. 262-266.

6. **Veličković P., Cucurull G., Casanova A., Romero A., Liò P., Bengio Y.** Graph Attention Networks // arXiv. 2017.

7. **Горчаков А.В.** Методы и алгоритмы идентификации фрагментов программных систем для формирования рекомендаций по повышению их быстродействия // Вестник Рязанского государственного радиотехнического университета. 2023. № 86. С. 96-109.

8. **Морошкин Н.А., Демидова Л.А.** Процесс трансляции регулярных выражений разных диалектов с оптимизацией промежуточных представлений // ИТ-Стандарт. 2024. № 4(41). С. 42-58.

9. **Zhou J., Cui G., Hu S., Zhang Z., Yang C., Liu Z., Wang L., Li C., Sun M.** Graph neural networks: A review of methods and applications // AI Open. 2020. № 1. С. 57-81.

10. **Kipf T.N., Welling M.** Semi-Supervised Classification with Graph Convolutional Networks // arXiv. 2016.

11. **Hamilton W.L., Ying R., Leskovec J.** Inductive Representation Learning on Large Graphs // arXiv. 2017.

12. **Xu K., Hu W., Leskovec J., Jegelka S.** How Powerful are Graph Neural Networks // arXiv. 2018.

13. **Li Y., Gu C., Dullien T., Vinyals O., Kohli P.** Graph Matching Networks for Learning the Similarity of Graph Structured Objects // arXiv. 2019.

14. **Oord A.v.d., Li Y., Vinyals O.** Representation Learning with Contrastive Predictive Coding // arXiv. 2018.

15. Regex101 [Электронный ресурс]. URL: https://regex101.com/ (дата обращения: 01.04.2025).

16. Документация к библиотеке PyTorch [Электронный ресурс]. URL: https://pytorch.org/docs/stable/index.html (дата обращения: 01.04.2025).

17. Документация к библиотеке PyTorch Geometric [Электронный ресурс]. URL: https://pytorch-geometric.readthedocs.io/en/latest/ (дата обращения: 01.04.2025).

18. **Kingma D.P., Ba J.** Adam: A Method for Stochastic Optimization // arXiv. 2014.

19. **Корячко В.П., Викулин С.Д., Волков А.В.** Применение методов кластеризации для анализа свойств материалов // Вестник Рязанского государственного радиотехнического университета. 2024. № 89. С. 77-84.

20. **Rousseeuw P. J.** Silhouettes: A graphical aid to the interpretation and validation of cluster analysis // Journal of Computational and Applied Mathematics. 1987. № 20. С. 53-65.

21. **Demidova L.A., Gorchakov A.V.** Fuzzy Information Discrimination Measures and Their Application to Low Dimensional Embedding Construction in the UMAP Algorithm // Journal of Imaging. 2022. № 8. С. 113.

22. **Kozma L., Voderholzer J.** Theoretical Analysis of Byte-Pair Encoding // arXiv. 2024.

UDC 004.85

AN APPROACH TO REGULAR EXPRESSION ANALYSIS USING GRAPH NEURAL NETWORKS AND CONTRASTIVE LEARNING

L. A. Demidova, Dr. in technical sciences, Full Professor, Professor at the Department of Corporate Information Systems, Institute of Information Technologies, MIREA – Russian Technological University, Moscow, Russia;
orcid.org/0000-0003-4516-3746, e-mail: demidova.liliya@gmail.com
V. E. Zhuravlev, Post-graduate Student at the Department of Corporate Information Systems, Institute of Information Technologies, MIREA – Russian Technological University, Moscow, Russia;
orcid.org/0009-0008-2942-0312, e-mail: vovcrane@mail.ru

The paper explores an approach to feature extraction from regular expressions using graph neural networks and contrastive learning. A novel method for constructing graph representations of regular expressions based on their textual form is proposed. The resulting graphs preserve both semantic and structural properties of original regular expressions. To analyze these graph representations, a machine learning model is introduced, leveraging graph neural network and global aggregation with attention mechanism. Model parameters are optimized using contrastive learning in self-supervised paradigm, where similar graphs are generated automatically through random augmentations. The experiments utilize a dataset of several thousand regular expressions collected from Regex101 website. The final model, trained on a dedicated training subset, is evaluated based on the quality and interpretability of vector representations for regular expressions it produces. To assess this, clustering is performed on validation subset, demonstrating high quality of feature extraction from regular expressions and confirming the effectiveness of graph neural networks and contrastive learning.

Keywords: *regular expressions, machine learning, feature extraction, representation learning, graph neural networks, attention mechanism, contrastive learning, clustering, k-means.*

DOI: 10.21667/1995-4565-2025-92-106-120

References

1. **Kozlov S.V., Svetlakov A.V.** Primenenie regulyarnykh vyrazheniy dlya obrabotki tekstovykh dannykh. *International Journal of Open Information Technologies*. 2022, no. 9. (in Russian).

2. **Panamareva O.N., Grebenshchikov D.A., Sukharev D.A.** Mnogotselevaya fil'tratsiya teksta s ispol'zovaniem regulyarnykh vyrazheniy. *Izvestiya Tul'skogo gosudarstvennogo universiteta. Tekhnicheskie nauki*. 2024. no. 7, pp. 284-288. (in Russian).

3. **Wang P., Brown C., Jennings J.A., Stolee K.T.** Demystifying regular expression bugs: A comprehensive study on regular expression bug causes, fixes, and testing. *Empirical Software Engineering*. 2021, no. 27.

4. **Kozik R., Choraś M., Renk R., Hołubowicz W.** Modelling HTTP Requests with Regular Expressions for Detection of Cyber Attacks Targeted at Web Applications. *International Joint Conference SOCO' 14-CISIS'14-ICEUTE'14*. 2014, pp. 527-535.

5. **Yan T., Yoo D., Berardini T.Z., Mueller L.A., Weems D.C., Weng S., Cherry J.M., Rhee S.Y.** PatMatch: a program for finding patterns in peptide and nucleotide sequences. *Nucleic Acids Research*. 2005. no. 33, pp. 262-266.

6. **Veličković P., Cucurull G., Casanova A., Romero A., Liò P., Bengio Y.** Graph Attention Networks // *arXiv*. 2017.

7. **Gorchakov A.V.** Metody i algoritmy identifikatsii fragmentov programmnykh sistem dlya formirovaniya rekomendatsiy po povysheniyu ikh bystrodeystviya. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta*. 2023, no. 86, pp. 96-109. (in Russian).

8. **Moroshkin N.A., Demidova L.A.** Protsess translyatsii regulyarnykh vyrazheniy raznykh dialektov s optimizatsiey promezhutochnykh predstavleniy. *IT-Standart*. 2024, pp. 4(41), pp. 42-58. (in Russian).

9. **Zhou J., Cui G., Hu S., Zhang Z., Yang C., Liu Z., Wang L., Li C., Sun M.** Graph neural networks: A review of methods and applications. *AI Open*. 2020, no. 1, pp. 57-81.

10. **Kipf T.N., Welling M.** Semi-Supervised Classification with Graph Convolutional Networks. *arXiv*. 2016.

11. **Hamilton W. L., Ying R., Leskovec J.** Inductive Representation Learning on Large Graphs. *arXiv*. 2017.

12. **Xu K., Hu W., Leskovec J., Jegelka S.** How Powerful are Graph Neural Networks. *arXiv*. 2018.

13. **Li Y., Gu C., Dullien T., Vinyals O., Kohli P.** Graph Matching Networks for Learning the Similarity of Graph Structured Objects. *arXiv*. 2019.

14. **Oord A.v.d., Li Y., Vinyals O.** Representation Learning with Contrastive Predictive Coding. *arXiv*. 2018.

15. Regex101 [Electronic resource]. *URL: https://regex101.com/* (accessed: 01.04.2025).

16. PyTorch documentation [Electronic resource]. URL: *https://pytorch.org/docs/stable/index.html* (accessed: 01.04.2025).

17. PyTorch Geometric documentation [Electronic resource]. URL: *https://pytorch-geometric.readthedocs.io/en/latest/* (accessed: 01.04.2025).

18. **Kingma D.P., Ba J.** Adam: A Method for Stochastic Optimization. *arXiv*. 2014.

19. **Koryachko V.P., Vikulin S.D., Volkov A.V.** Primenenie metodov klasterizatsii dlya analiza svoystv materialov. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta.* 2024, no. 89, pp. 77-84. (in Russian).

20. **Rousseeuw P.J.** Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*. 1987, no. 20, pp. 53-65.

21. **Demidova L.A., Gorchakov A.V.** Fuzzy Information Discrimination Measures and Their Application to Low Dimensional Embedding Construction in the UMAP Algorithm. *Journal of Imaging*. 2022, no. 8, pp. 113.

22. **Kozma L., Voderholzer J.** Theoretical Analysis of Byte-Pair Encoding. *arXiv*. 2024.