УДК 004.8, 519.178
DOI: 10.25686/978-5-8158-2474-4-2025-472-486

Метод идентификации неявных связей в решении аналитических задач

И. Д. Соколов, К. В. Ионкина, М. С. Улизко, Е. В. Антонов, Е. Н. Бажанова, А. А. Артамонов
Национальный исследовательский ядерный университет «МИФИ», Москва, Россия

Аннотация. Инструменты по визуализации позволяют преобразовать большие массивы данных в понятные графики для пользователя. В данной работе рассматривается разработка инструмента визуализации графов для выявления неявных связей между информационными объектами. В статье представлена реализация метода для выявления неявных связей между объектами и разработка инструмента для построения графовой визуализации, позволяющая пользователю взаимодействовать с графом посредством фильтрации. Реализованы функциональные возможности и предложен специализированный язык запросов для изменения вида узлов и ребер графа.

Разработанный инструмент и предложенный метод апробированы на двух реальных наборах данных: обнаружение потенциальных нарушений обязательств по ядерному нераспространению, выявление перспективных направлений научного сотрудничества организаций, подтвердив практическую значимость данного исследования.

Method of identification of implicit connections in solving analytical problems

I. D. Sokolov, K. V. Ionkina, M. S. Ulizko, E.V. Antonov, E. N. Bazhanova, A. A. Artamonov
National Research Nuclear University “MEPhI”, Moscow, Russia

Введение

В настоящий момент существует множество инструментов для визуализации данных. Однако каждый из них имеет определенные ограничения. Коммерческие программные решения могут быть недоступны из-за высокой стоимости, другие решения могут не обладать достаточной функциональностью или сложны в освоении. Например, приложение Gephi, широко используемое для визуализации графовых данных, требует от пользователя высокой степени подготовки, что обусловлено сложностью его интерфейса и многоступенчатым процессом построения графов. VOSviewer требует структуру и выполняет одну узконаправленную задачу (задача библиометрического анализа). В связи с этим возникает потребность в создании специализированного программного обеспечения, обладающего возможностями интеграции интерактивных механизмов фильтрации и методов выделения ключевых вершин и ребер для анализа больших объемов данных и выявления закономерностей, при этом не привязанного к определенной структуре данных.

Таким образом, необходимость в гибком и функциональном инструменте для анализа сложных данных становится особенно актуальной при исследовании неоднородных объектов, где важно выявлять скрытые связи и зависимости.

Для определения связи между объектами A и B необходимо выделить характеристики этих объектов и выявить общие признаки, которые и будут указывать на взаимосвязь между исходными объектами. Пусть $C -$ объект, такой что C имеет общие значения характеристик с объектом A и общие значения характеристик с объектом B. В этом случае можно предположить, что объекты A и B связаны через объект C. Таким образом, объект C становится общим признаком для объектов A и B, что позволяет связать эти два объекта между собой.

В статье «Нахождение скрытых зависимостей между объектами на основе анализа больших массивов библиографических данных» [1] рассматривается задача определения тематической близости журналов и конференций, используя несколько подходов определения близости автора к научной области. Автор работы рассматривает подход построения графов соавторства статей, в котором рассчитывается вес ребра, соединяющего несколько журналов. Предполагается, что если один и тот же автор имеет публикацию в разных журналах, то это может свидетельствовать о тематической близости журналов.

В статье «Методы анализа гетерогенных данных для построения социального профиля» [2] рассматривается выявление неявных связей для построения социального профиля. В качестве метода предлагается использовать графовый подход для нахождения неявных связей.

Указанные работы доказывают целесообразность использования графовой визуализации при выявлении взаимосвязей между объектами. Для определения взаимосвязи объектов аналитик должен выбрать интересующие его характеристики в рамках рассматриваемого объекта и указать, как связаны сущности внутри, определяя правило построения графа.

Рассмотрим существующие методы выявления связей между объектами. В работе [1] рассматривался метод графа соавторства. Данный метод применяется в основном для определения тематической близости журналов и конференций за счет построения графа. Метод имеет ограничение, связанное с масштабируемостью, так как применяется только для одного вида задач.

Рассмотрим метод корреляции (определения рассмотрены в книге [3]). Данный метод определяет взаимосвязь между двумя переменными, что позволяет выявить зависимость изменения одной переменной от другой. Заметим, что хотя метод и не предполагает наличие структуры, но может работать только с одним типом данных.

В статье [4] представлено описание методов «Выявление причинности» и «Оценка причинности», позволяющих выявить причинно-следственную связь между рассматриваемыми объектами. Данные методы сконцентрированы на оценке воздействия одного элемента на другой и предполагают наличие причинности этой взаимосвязи. Однако в рамках системного анализа применение их нецелесообразно, так как они нацелены не на структуру объекта, а на причину и следствие влияния одного объекта на другой.

В статьях [5] и [6] рассмотрены использования подходов семантического анализа и построения онтологий. Эти два метода позволяют построить модель, которая описывает структуру рассматриваемой области, ее свойства и внутренние связи, а также представляет данную область в виде графа. Указанные методы решают задачу взаимосвязи объектов, но они привязаны к заранее известной структуре данных.

Постановка проблемы

Рассмотренные выше методы позволяют решать поставленные задачи на выявление связей между объектами, но требуют либо определенной структуры входных данных, либо определенного типа таких данных. В рамках настоящей работы необходимо решить проблему построения графа на входных данных различной структуры и типов, а также выявления явных и неявных связей между объектами, поскольку представленные выше методы такую задачу решить не позволяют. В ходе проведенного обзора решено разработать собственный метод для выявления явных и неявных связей независимо от структуры и типа входных данных и разработать инструмент для визуализации данных при различной структуре данных.

Теория

Метод идентификации связей между объектами

Пусть $N = \{obj = (a_1, a_2, \dots, a_M) | a_i - \text{характеристика объекта } obj, i = 1,2, \dots, M\}$ – множество объектов с характеристиками $a_1, a_2, \dots, a_M$. Каждому объекту $obj = (a_1, a_2, \dots, a_M) \in N$ поставим в соответствие граф $G =< V, E >$, где множество вершин $V = \{v_1, v_2, \dots, v_T\}, T \geq M$, составляют значения характеристик $a_1, a_2, \dots, a_M$ объекта obj, и множество ребер $E = \{(v_i, v_j) \mid \exists\, obj \in N : : obj(a_i) = v_i \text{ и } obj(a_j) = v_j\}$, то есть две вершины $v_i, v_j \in V$ соединены ребром, если найдется объект $obj = (a_1, a_2, \dots, a_M) \in N$, такой что значение характеристики a_i объекта obj равно v_i и значение характеристики a_j объекта obj равно v_j, где характеристики a_i и a_j задаются аналитиком.

Каждой вершине из множества V поставим в соответствие ее тип – характеристику из множества $\{a_1, a_2, \dots, a_M\}$. Тогда множество вершин $V = K_1 \cup K_2 \cup \dots \cup K_M$, где $K_i = \{v_j \in V | obj(a_i) = v_j\}$ – множество вершин типа a_i, $i = 1,2, \dots, M$. Если вершина $v_j \in K_i$, то для удобства будем обозначать ее $v_j^{K_i}$, $j \in \{1,2, \dots, T\}, i \in \{1,2, \dots, M\}$. Таким образом, граф $G =< V, E >$ можно представить в виде

гетерогенного графа $G =< K_1 \cup ... \cup K_L, E >$, где $L \in \{1,2,...,M\}$, при этом если $v_i, v_j \in K_W$, то $(v_i, v_j) \notin E$ (характеристики и их количество L задает аналитик).

Пусть K_L и K_P – множества вершин типа a_L и a_P соответственно. Приведем следующие определения:

Явная связь между вершинами $v_i \in K_L$ и $v_j \in K_P$ существует тогда и только тогда, когда есть ребро $(v_i, v_j) \in E$.

При построении графа, согласно правилу построения, между вершинами разных типов образуется явная связь. В таком случае между вершинами одного типа можно определить неявную связь.

Неявная связь между вершинами $v_i \in K_L$ и $v_j \in K_L$ существует тогда и только тогда, когда имеется вершина $v_x \in K_P$, такая что имеется ребро $(v_i, v_x) \in E$ и ребро $(v_j, v_x) \in E$, где $L \neq P$.

Явная связь порядка n между вершинами $v_i \in K_L$ и $v_j \in K_P$ существует тогда и только тогда, когда есть путь $v_i \to v_j$ длины n, не содержащий вершин одинаковых типов, причем $n \in \mathbb{N}$ – наименьшее натуральное число с таким свойством.

Неявная связь порядка n между вершинами $v_i \in K_L$ и $v_j \in K_L$ существует тогда и только тогда, когда имеется вершина $v_x \in K_P$, такая что существует явная связь $(v_i, v_x) \in E$ и $(v_j, v_x) \in E$, порядок хотя бы одной из которой равен n.

Очевидная неявная связь – неявная связь между вершинами $v_i \in V$ и $v_j \in V$, такая что существует $obj \in N$, значение характеристики a_i которого содержит v_i и v_j . В случае, если рассматриваются связи между объектами одного типа, очевидная неявная связь превращается в явную связь.

Стоит отметить, что данные определения справедливы для неориентированного графа, так как в ориентированном графе присутствует направление от одной вершины к другой и отсутствует потребность в типах вершин.

Алгоритм построения графа на различной структуре входных данных

В зависимости от поставленной задачи структура входных данных может быть различной. Каждый объект может отличаться по наименованию полей, типу полей и разной структурой вложенности. В таком случае рассмотрены четыре вида связи полей между собой внутри любой структуры:

- *Вид 1*: Связывание полей, не имеющих общих предков;
- *Вид 2*: Связывание полей, находящихся на одном уровне, имеющих общего предка;
- *Вид 3*: Связывание полей, находящихся на разном уровне, имеющих общего предка;
- *Вид 4*: Связывание полей одного поля.

```
{
  "Title": "article 1",
  "author": {
    "full_name": "Author's name",
    "affiliation": {
      "name": "National Resaerch Nuclear University MEPhI",
      "country": "Russian Federation"
    },
    "published_year": 2021,
    "keyword": "React"
  }
}
```

Рис. 1. Пример файла с обычной вложенностью

Рассмотрим файл со следующей структурой (рис. 1). Для получения полей из такой структуры данных и их последующего связывания будет достаточно обратиться напрямую к полю. Если интересующее поле находится на n-ом уровне вложенности, то данное поле получает предка, который описывает путь получения данных по полю. Предком считается предшествующее поле в данных. Если структура данных представляет сильную вложенность, то вводится понятие ближайшего предка. Ближайший предок – поле в данных, при обращении к которому выводится интересующее поле. Для одного поля может существовать только один ближайший предок, таким образом каждое поле будет

иметь путь, состоящий из предков и одного ближайшего предка. Ближайший предок может относиться к нескольким полям, например поле affiliation является ближайшим предком для «name» и «country», по которым можно получить название аффилиации и страну.

Общим предком считается поле, по которому необходимо обратиться для получения искомых данных. Стоит отметить, что ближайший предок может быть и общим, при условии, что уровень вложенности равен одному. В противном случае общим предком будет считаться поле, находящиеся на верхнем уровне вложенности, а ближайший предок – на предпоследнем уровне вложенности.

Рассмотрим файл с вложенной структурой. На рисунке 2 представлен пример данных, в котором по одному полю можно получить список данных с одинаковыми полями. В этом случае обратиться напрямую к полю и связать их не получится, так как при обращении к предку выводится список значений. Для такого случая необходимо получить данные из списка и работать с элементами списка по отдельности, используя один и тот же алгоритм связи. В таком случае реализовывается декартово произведение элементов.


```
{
  "Title": "article_1",
  "author": [
    {
      "full_name": "Author's name",
      "affiliation": [
        {
          "name": "National Research Nuclear University MEPhI",
          "country": "Russian Federation"
        }
      ]
    },
    {
      "full_name": "Another Authors's name",
      "affiliation": [
        {
          "name": "National Research Nuclear University MEPhI",
          "country": "Russian Federation"
        },
        {
          "name": "Moscow State University",
          "country": "Russian Federation"
        }
      ]
    }
  ],
  "published_year": 2021,
  "keyword": [
    "React",
    "Python"
  ]
}
```


Рис. 2. Пример файла со вложенностью данных

Для таких структур ниже рассмотрены различные примеры связи полей.

Связывание полей, не имеющих общих предков

Из-за того, что поля не имеют общих предков, для связи таких полей будет достаточно связать их, используя путь до интересующего поля. При структуре, изображенной на рисунке 1, для связывания доступны следующие пары полей:

- Title – author.full_name;
- Title – author.affiliation.name;
- Title – author.affiliation.country;
- Title – published_year;
- и другие варианты, при которых узлы без общих предков будут связываться.

Если значением одного из выбранных полей будет список, то для связывания такого рода необходимо произвести декартово произведение. Например, для связи авторов с ключевыми словами следует сначала пройтись по списку, получаемому по ключу “author”, получить данные по полю “full_name”, после пройтись по полу “keyword”, то есть произвести декартово произведение выбранных полей. Таким образом, получатся следующие пары узлов:

- Author’s name – React;
- Author’s name – Python;
- Another Author’s name – React;
- Another Author’s name – Python;
- и другие, при которых узлы без общих предков будут связываться.

Связывание полей, находящихся на одном уровне, имея общего предка

Данный вид позволяет связать поля, находящиеся на одном уровне вложенности, например название аффилиации и страна аффилиации. Возьмем структуру, представленную на рисунке 2.

Для связывания полей, находящихся на одном уровне, необходимо получить данные общего ближайшего предка и соединить их.

Данная связь необходима для того, чтобы связать поля, относящиеся к определенному классу данных, например: полная информация об авторе, аффилиации и т. д. При этом обеспечивается связь только внутренних данных, что позволяет сделать поэлементное соединение, то есть соединение, основанное на позиции элементов внутри данных, где соединение происходит по правилу $[[a],[b]],[[x],[y]] \rightarrow [[a,b],[x,y]]$.

При структуре данных (см. рис. 2) возможно связать следующую пару полей:

- author.affiliation.name – author.affiliation.country.

Связывание полей, находящихся на разном уровне, но имея общего предка

Данный вид связи представляет собой комбинацию первого и второго вида, при котором необходимо получить данные в рамках одного предка, как при использовании второго вида связи. После получить информацию по второму полю и произвести декартово умножение как для первого вида.

Предположим, пользователь хочет связать следующие поля: author.full_name и author.affiliation.country. Для этого нужно привести данные к одному уровню и связать поля. При структуре данных (рис 2) возможно связать следующие пары полей:

- author.full_name – author.affiliation.country,
- author.full_name – author.affiliation.name.

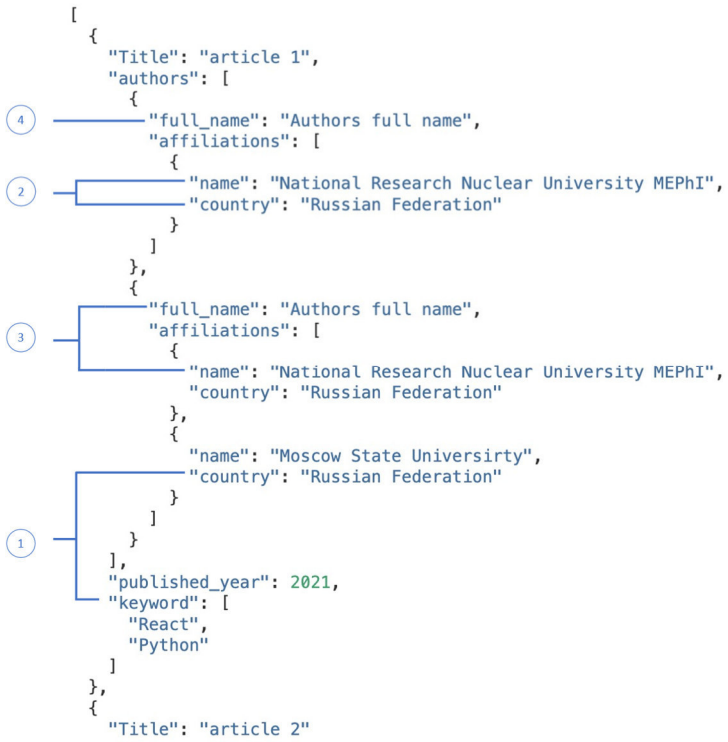


Рис. 3. Пример структуры данных

Связывание элементов одного поля

Этот вид представляет собой частный случай первого вида связи, при котором выбраны два одинаковых поля. Такой вид может использоваться при связывании элементов в рамках одной записи. Например, рассмотрим поле “full_name”. В рамках одного объекта (в нашем случае статьи Title 1) все авторы будут связаны между собой. Таким образом получаются узлы со следующими связями:

- Author 1 – Author 1 (для первой статьи);
- Author 1 – Author 2 (для второй статьи);
- Author 2 – Author 3 (для третьей статьи).

При рассмотренных видах связи можно сформировать следующие варианты связывания полей внутри одной структуры данных. Каждый вид обозначен своим номером: 1 – первый вид связи, 2 – второй вид связи, 3 – третий вид связи, 4 – четвертый вид связи (рис. 3).

В зависимости от вида связи будет изменяться алгоритм связывания (рис. 4). Для каждого вида реализован программный код на основе предложенного алгоритма, который позволяет проходить по всей структуре и получать необходимые данные по полям, заранее приводя их в структуру для построения графа.

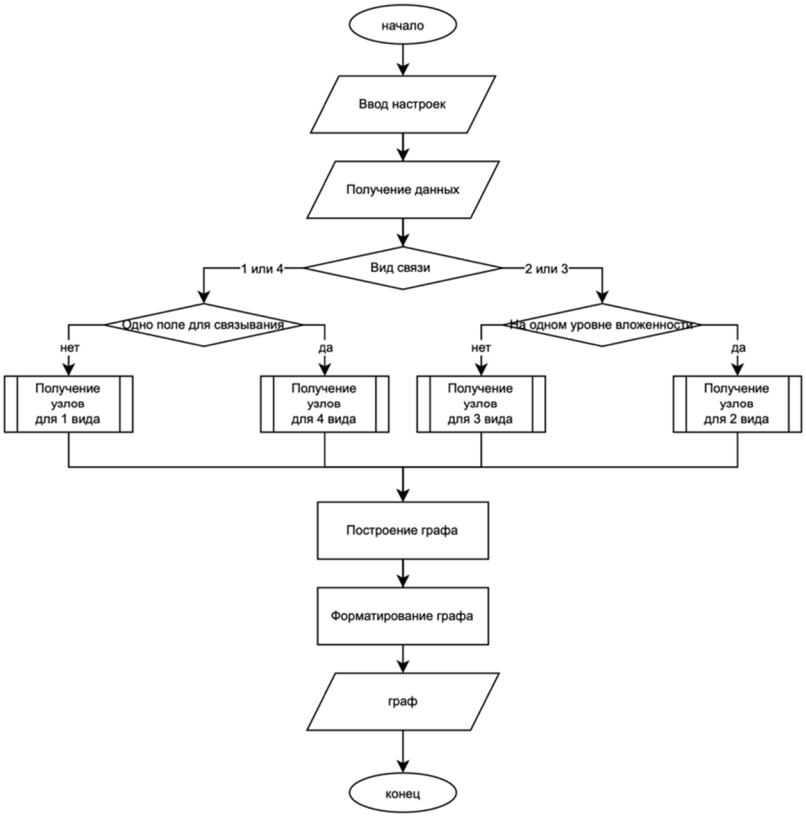


Рис. 4. Алгоритм построения графа

3. Проектирование и разработка инструмента для визуализации графа

Разработанный инструмент представляет собой клиент-серверную архитектуру. В данном случае клиент задает настройку графа, а сервер выполняет запрос в базу данных и преобразовывает данные в структуру для построения графа.

Инструмент состоит из трех компонентов (рис. 5): графического интерфейса, программного интерфейса и базы данных. Графический интерфейс позволяет пользователю взаимодействовать с программным интерфейсом посредством использования определенных фильтров. В свою очередь программный интерфейс взаимодействует с базой данных. В качестве хранения выбран ElasticSearch. При выгрузке данных программный интерфейс преобразует их в формат для построения графа.

Программный интерфейс реализован на языке программирования Python, графический интерфейс – ReactJS.

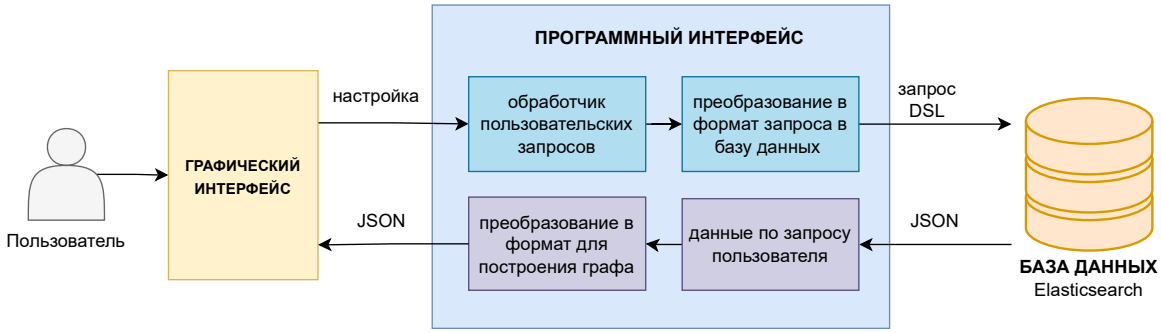


Рис. 5. Архитектура инструмента для построения графа

4. Функциональные возможности инструмента

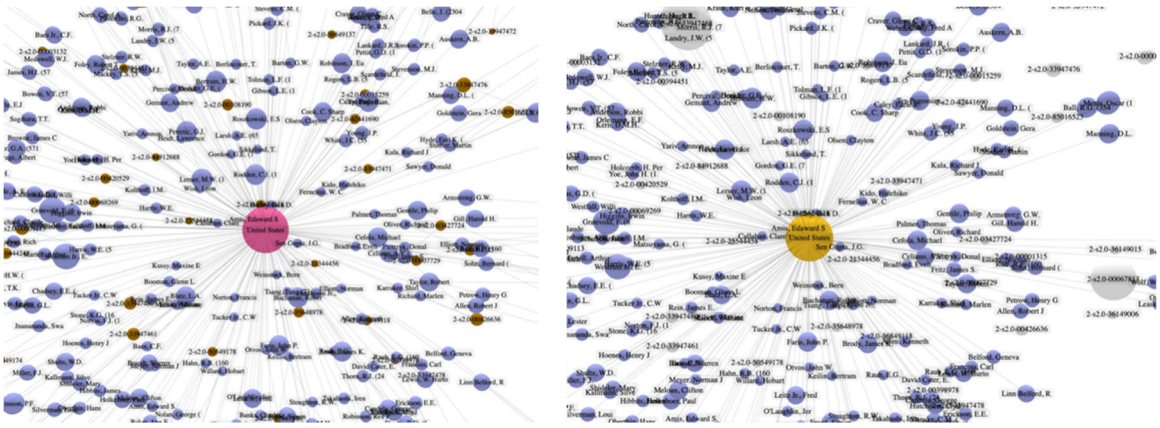
Разработанный инструмент обладает функциональными возможностями, направленными на поддержку аналитического процесса посредством акцентирования внимания пользователя на определённых сегментах графа. В рамках этого реализованы следующие ключевые механизмы:

- выделение интересующей вершины;
- сброс выделения;
- подсвечивание вершин;
- фильтрация графа;
- кастомизация узлов.

Все перечисленные механизмы вместе создают гибкую среду, в которой аналитик может оперативно переключаться между глобальным обзором и детальным исследованием любых сегментов графа.

Выделение вершины

Данный механизм предназначен для выделения интересующей вершины графа и всех сопутствующих связей (рис. 6). Все не связанные с выбранной вершины перекрашиваются в серый цвет, что позволяет сконцентрироваться на отдельном рассматриваемом сегменте. При выборе вершины ее цвет меняется, удерживая внимание пользователя на выбранном объекте. Данная функциональная возможность позволит пользователю отсеять все вершины, не представляющие интерес на текущем этапе, даст возможность сконцентрировать внимание на вершинах и их связях с другими информационными объектами.



а б

Рис. 6. Выделение вершины графа: *а*) состояние графа до выделения вершины; *б*) состояние графа после выделения выбранной вершины

При выборе узла в блоке дополнительной информации отобразится сводка по узлу, то есть выведется название узла, его вес и группа (рис. 7).

Рис. 7. Пример отображения дополнительной информации по узлу

Сброс выделения

Необходимость сбрасывания выделения вершины обусловлена возможностью выбора других элементов графа без остаточных эффектов от предыдущего выделения. При нажатии кнопки «сброс выделения» все вершины перекрашиваются в свой первоначальный цвет. Также блок дополнительной информации уберет сводку по выбранному узлу.

Подсвечивание вершин

Для акцентирования внимания аналитика на определенных узлах добавляется подсвечивание вершин графа (рис. 8). Данная функциональная возможность позволит привлечь внимание аналитика к определенной вершине графа по заданному условию для прослеживания связи с объектами. Подсветка реализовывается с использованием мигающей сферы.

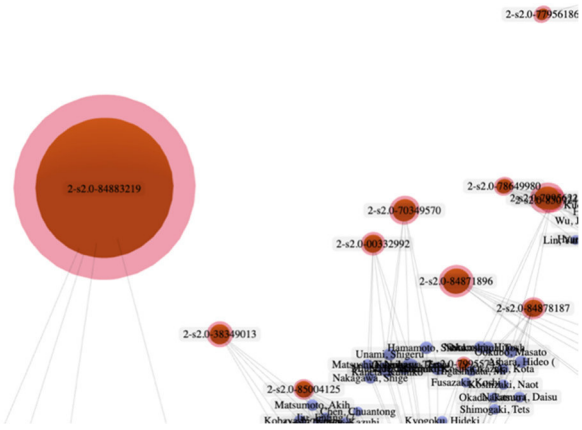


Рис. 8. Сфера для акцентирования внимания пользователя

Кастомная настройка узлов

Пользователь для интересующих его узлов может задать цвет и группу, к которой узлы будут относиться. Цвет для той или иной группы узлов генерируется автоматически (рис. 9). Также

пользователю необходимо задать способ связи групп в отведенном поле «Отношения между полями». Выбранные поля будут отмечаться цветом той группы, к которой они принадлежат. При необходимости пользователь может настроить для группы узлов наличие подписи и мигающей сферы.

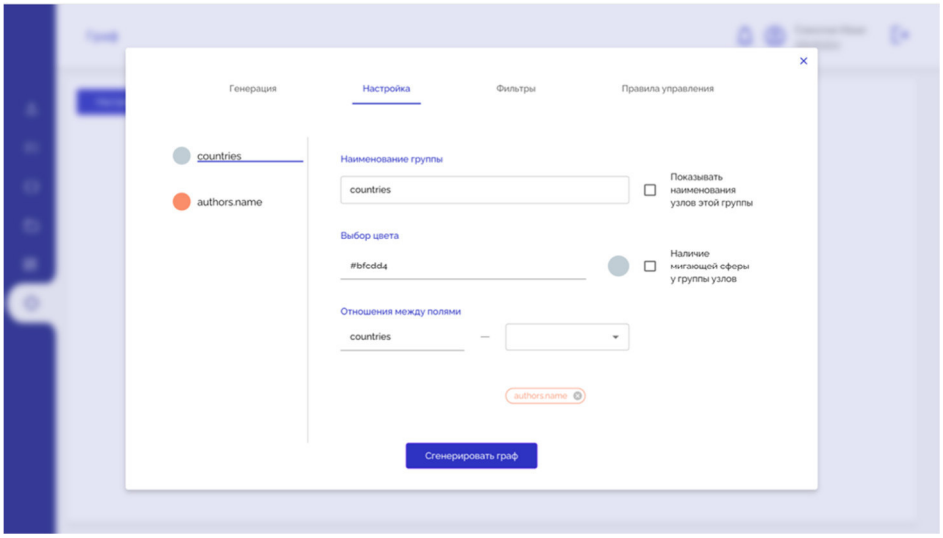


Рис. 9. Настройка узлов графа по выбранным полям

Фильтрация данных

Данный механизм позволяет осуществлять фильтрацию графа для уменьшения количества данных. Фильтрация производится с использованием языка запросов Domain Specific Language (DSL) на основе формата JSON для составления запросов. Для поиска определенного документа, загруженного в базу данных, необходимо обратиться к созданному полю посредством определенного синтаксиса языка запросов {query: {<тело запроса>}}.

Для эксплуатации необходимо наличие два способа фильтрации: фильтрация посредством выбора соответствующих полей, операторов и типа фильтрации; расширенная фильтрация, позволяющая писать запросы, используя язык запросов Domain Specific Language (DSL) (рис. 10).

Первый метод позволяет соединять выбранные фильтры логическими операторами (must, should, must not). Между собой фильтры соединяются логическим и, для учитывания всех введенных фильтров по интересующим полям. Например, фильтрация документов по странам и фильтрация документов по определенным ключевым словам будут соединяться между собой логическим «И», то есть в документе должны учитываться условия как первого фильтра, так и второго.

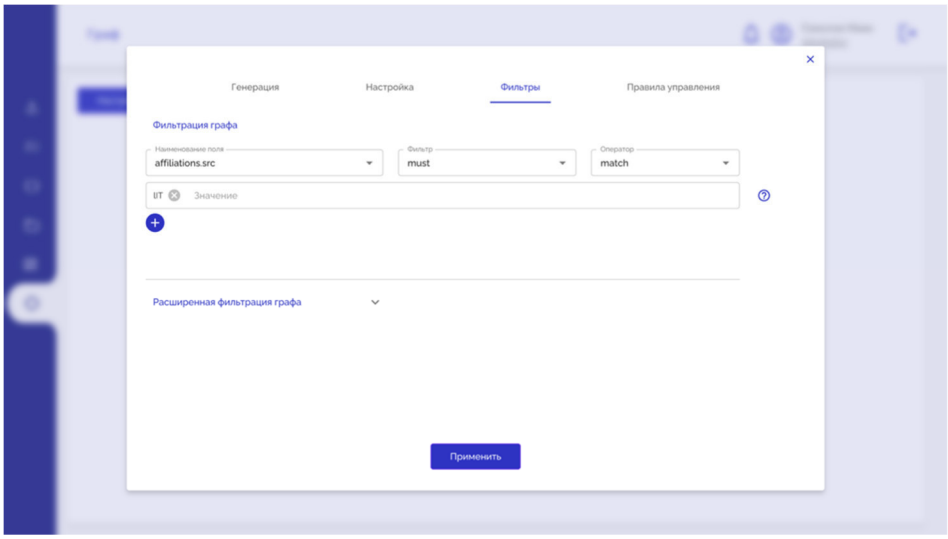


Рис. 10. Фильтрация графа по запросу в Elasticsearch

Правила для изменения выбранных узлов

Для решения аналитических задач предложен язык формирования запросов (правил). Рассмотрим случай, при котором пользователю необходимо выделить из определенной группы узлов узел или узлы, удовлетворяющие определенным условиям. Для этого реализованы правила, на основе которых пользователь может изменить значения узла (например, изменить вес узла, цвет, группу, наименование и подсвечивание). Для этого реализован специализированный язык запросов:

Базовая структура правил

field_type : <values> {conditions = conditions} [change_pattern = change_pattern]

Здесь

- field_type синтересующее поле, узлы которого нужно рассматривать;
- <values> – список узлов, которые необходимо рассмотреть. Для этого списка узлов будет применяться изменение. Узлы разделяются запятыми;
- {conditions = conditions} – список условий, при удовлетворении которых будет применяться изменение. Новое условие разделяется запятой и связывается логическим И (документ должен выполнять все условия);
- [change_pattern = change_pattern] – список изменеий.

Изменению подлежат следующие значения:

- val – вес узла (int);
- name – наименование узла (str);
- label – лэйбл узла (str);
- label_vis – наличие лэйбла у узла (boolean);
- color – цвет узла (str);
- highlited – наличие мигающей сферы (boolean);
- group – группа узла (str).

Для присвоения одному из значений определенного типа данных необходимо перед этим значением указать сам тип данных. Реализованы следующие типы данных:

int() – числовые;

float() – десятичные или числа с плавающей точкой (0.5/1.0);

bool() – логические (True/False);

str() – текстовые. Если в текстовых данных присутствуют запятые, то нужно заключить их в слэш (/, /).

Если в поле field_type указано значение all – {conditions = conditions} и [change_pattern = change_pattern] будут применяться ко всем полям и узлам.

Если в поле <values> указано <*> – {conditions = conditions} и [change_pattern = change_pattern] будут применяться к узлам по выбраному полю field_type.

Для указания {conditions = conditions} необходимо точно указать поле, по которому нужно провести сравнение (доступны следующие виды сравнения: =, =>, >, <=, <). Каждое условие связано с предыдущим логическим «И». Пример написания условия:

{condition 1 = value 1, condition 2 <= value 2}

Здесь condition 1/ condition 2 – поля, по которым нужно провести условие; value 1/ value 2 – значение для сравнения.

Для указания [change_pattern = change_pattern] необходимо указать изменение. Пример написания изменения:

[color = user_color, val = int(user_val)]

Здесь user_color – цвет, который задал пользователь; user_val – вес, который задал пользователь.

Пример реализованных правил:

author.gender: <female> [color = red];
countries: <US, RU> [color = blue, group = countries];
eid: <> {probability >= float(0.5)} [val = int(10), highlited = bool(True)]*

В графическом интерфейсе для того, чтобы использовать правила, реализована отдельная вкладка, в которую необходимо вводить правило (рис. 11).

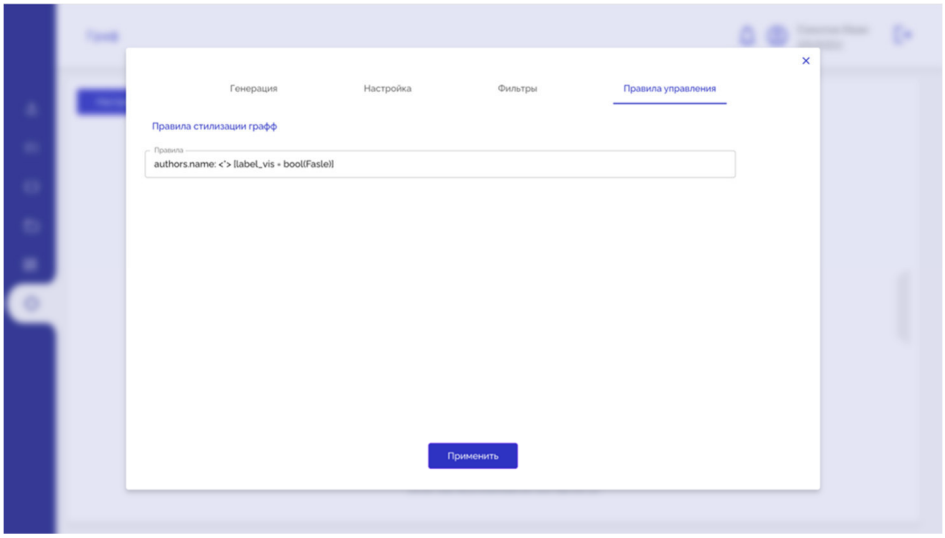


Рис. 11. Пример реализации правил для настройки узлов графа

Результаты экспериментов

Разработанный инструмент используется для решения научно-технических задач. В рамках апробации инструмента рассмотрены две задачи: обнаружение нарушений обязательств по ядерному нераспространению, визуальный анализ публикационной активности научной организации.

5. Обнаружение нарушений обязательств по ядерному нераспространению

Апробация системы в данной задаче проходит на 5 280 публикациях по ядерным технологиям. Для построения графа выбраны следующие поля: countries – страны (зеленые узлы), authors.full_name – полное имя автора (голубые узлы), eid – идентификатор статьи (коричневые узлы). Полученный граф отображен на рисунке 12.

Для акцентирования внимания на ядерных державах узлы, представляющие страны «ядерной пятёрки», окрашены в зелёный цвет, тогда как все остальные страны отображаются красным.

При исследовании графа обнаружено, что у США есть совместные публикации с двумя странами не из ядерной пятерки (Австралия и Норвегия), и по ним производилась фильтрация. Для уменьшения количества публикаций производилась фильтрация для статей США, в которых должны были встречаться Австралия или Норвегия.

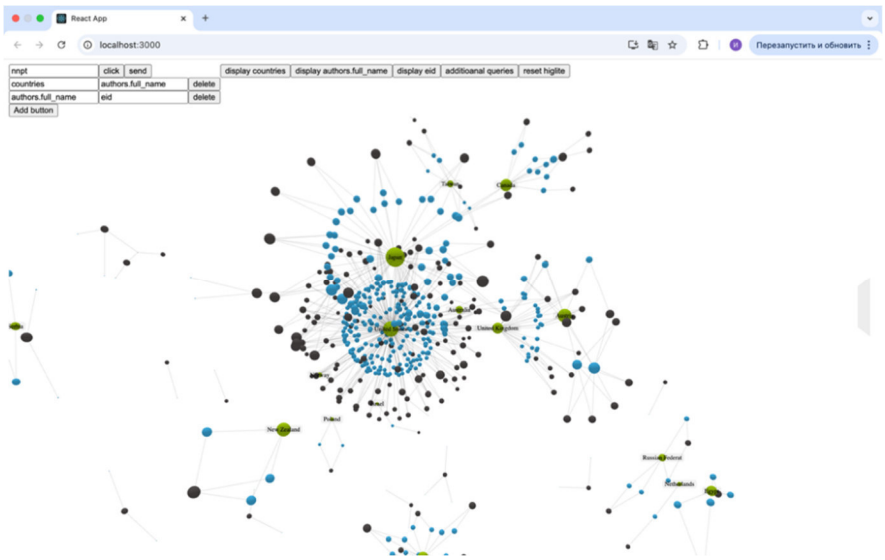


Рис. 12. Граф взаимосвязи авторов, стран и статей

После применения фильтрации графа обнаружено, что с США писали еще Германия и Дания (рис. 13). Таким образом, можно выделить коллаборационные статьи, представляющие интерес для эксперта.

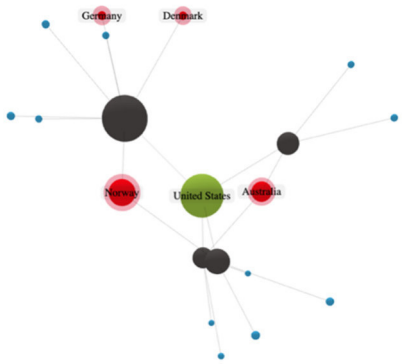


Рис. 13. Фильтрация графа по США

Также построен граф тематик исследований по странам (рис. 14). Из графа можно выделить пересечение стран, которые пишут по нескольким тематикам (красное пересечение) и группы стран, которые работают в основном только по одной тематике (синее пересечение).

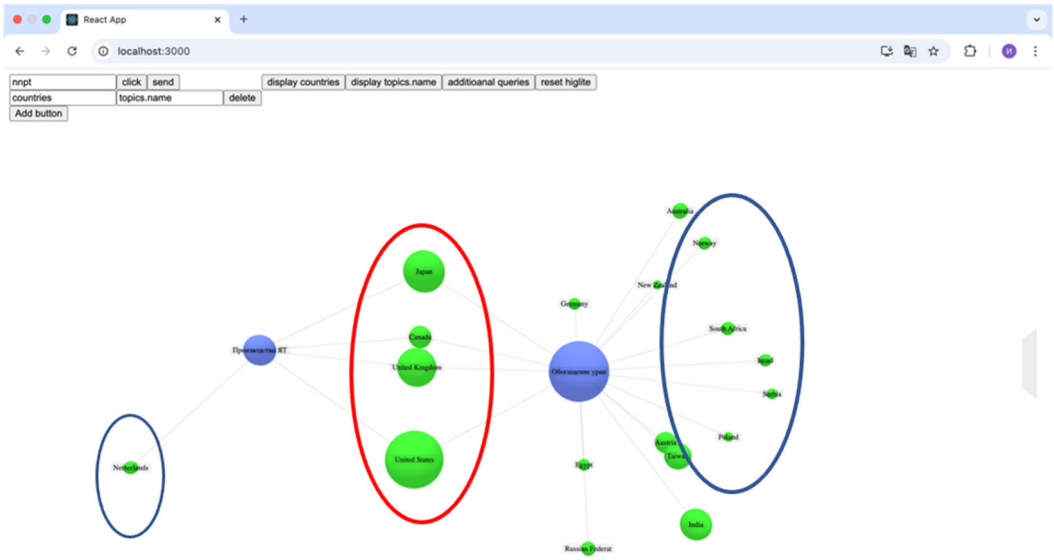


Рис. 14. Граф по тематикам исследований стран

6. Визуальный анализ публикационной активности научной организации

Для анализа структуры и тематической направленности публикационной активности Объединённого института ядерных исследований (ОИЯИ) использован метод визуализации на основе графовых моделей. В качестве данных применены метаданные 36 008 публикаций. На этих данных строится граф по следующим полям: affiliations.name (название аффилиации) – красные узлы и keyword (ключевые слова) – розовые узлы. В итоге реализован граф с 1470 узлами и 5291 ребром (рис. 15).

Полученный граф разделяется на несколько кластеров, которые описывают взаимосвязи организаций по определенным тематикам. На рисунке 16 представлены направления тематик ОИЯИ. По выделенным узлам можно выявить, с какими организациями ОИЯИ взаимодействует и по каким тематикам.

Для уменьшения количества данных производилась фильтрация по ОИЯИ. В итоге реализован граф из 1406 узлов и 5504 ребер (рис. 17).

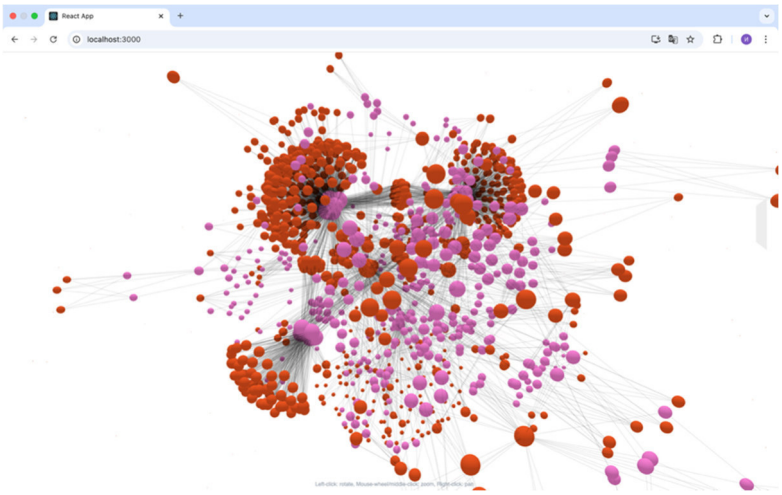


Рис. 15. Граф публикационной активности ОИЯИ

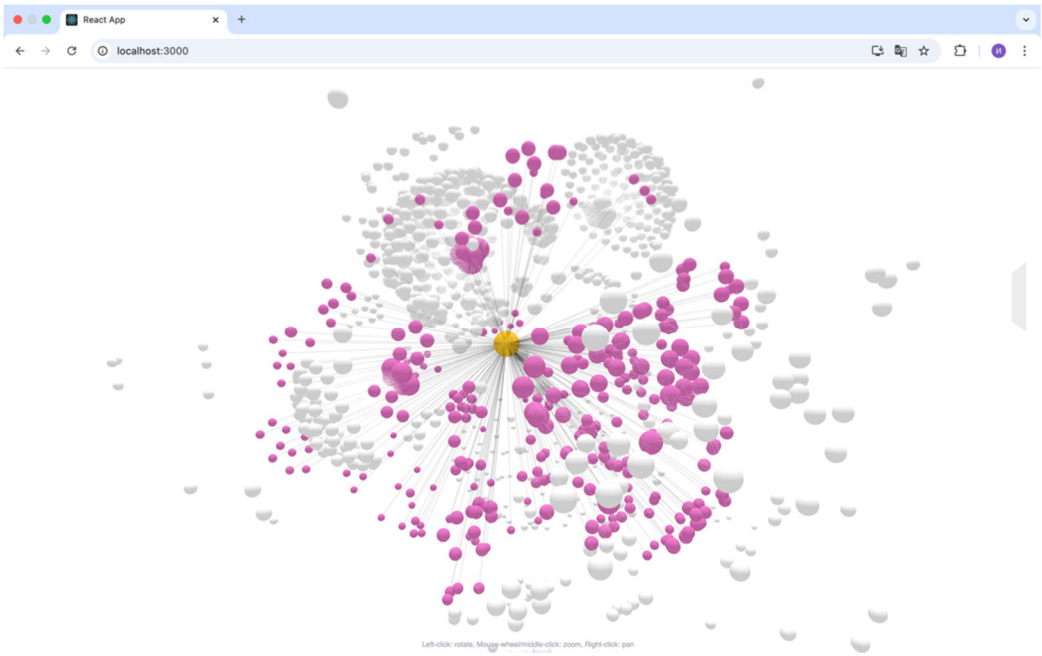


Рис. 16. Подсвечивание связей узла ОИЯИ

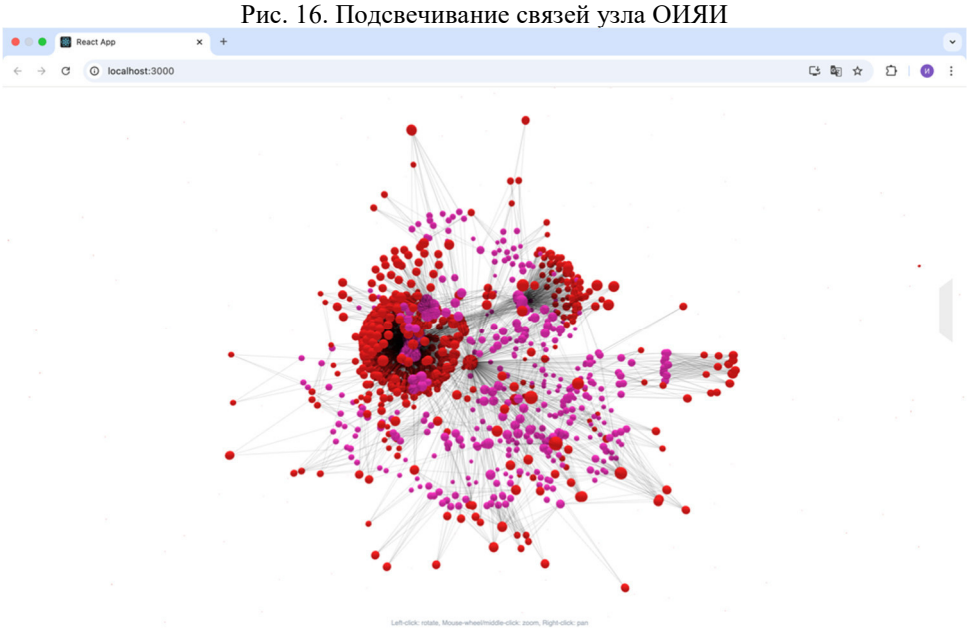


Рис. 17. Отфильтрованный граф по ОИЯИ

На полученном графе чётко прослеживаются связи ОИЯИ с внешними организациями. В частности, кластер, представленный на рисунке 18, посвящен тематике физики частиц и демонстрирует основные взаимодействия института в этой области.

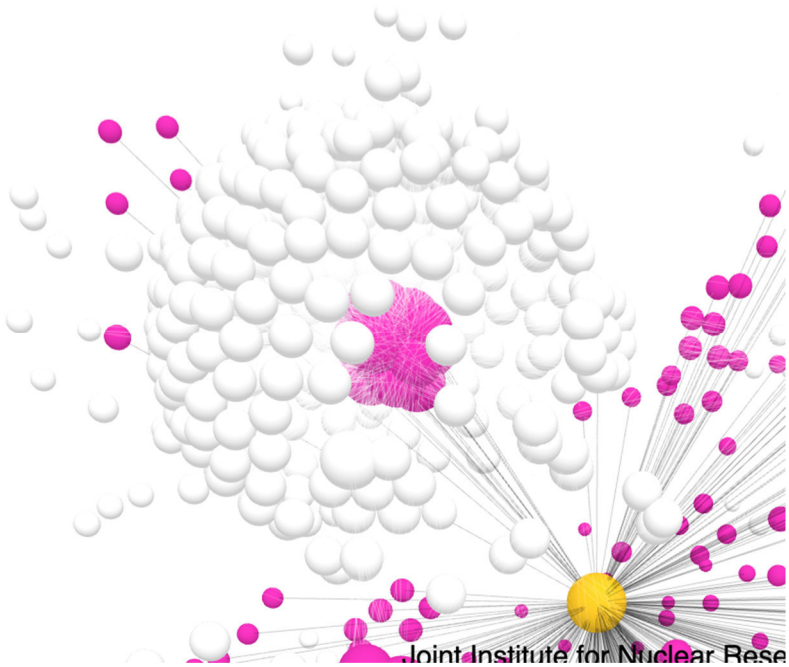


Рис. 18. Кластер тематики ОИЯИ, посвященный направлению физики частиц

Применение фильтрации по ключевым словам «jets» и «quark gluon plasma» позволяет детализировать эти связи и выделить конкретные организации-партнёры ОИЯИ в рамках данного научного направления (рис. 19).

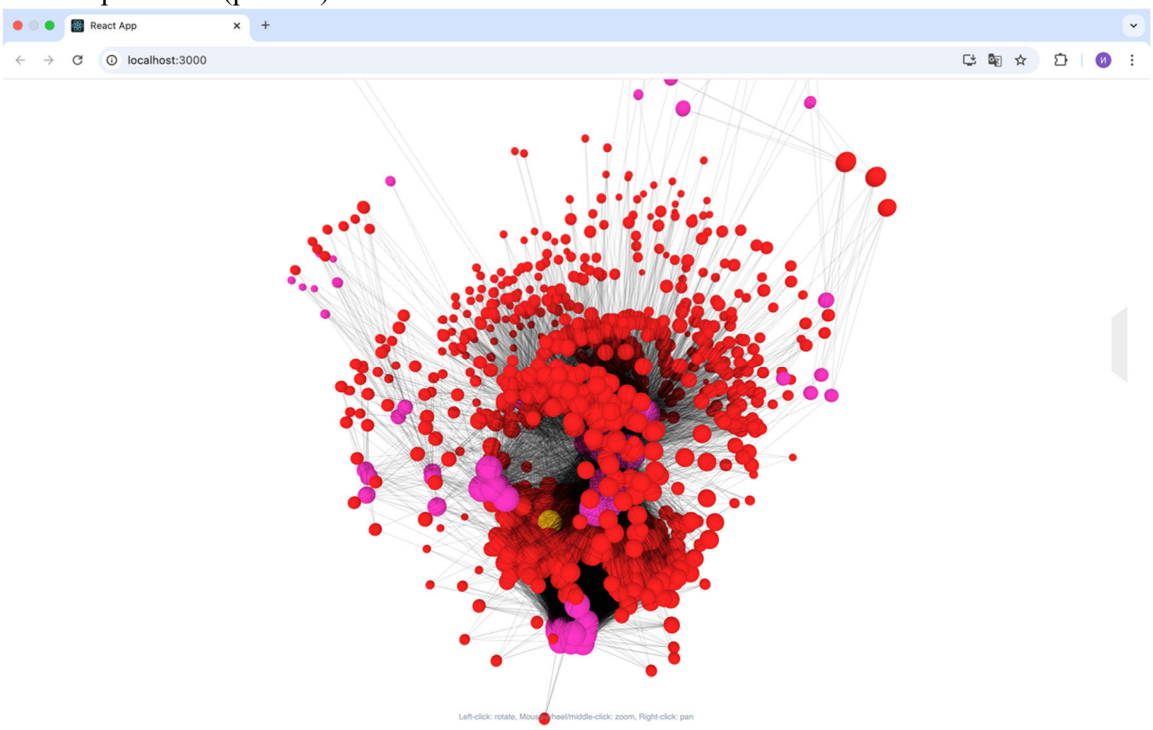


Рис. 19. Граф взаимосвязей между организациями по выбранной тематике (ОИЯИ – желтый узел)

Обсуждение результатов

Разработанный метод идентификации явных и неявных связей между объектами, а также реализованный инструмент построения графовой модели продемонстрировали высокую

эффективность при решении аналитических задач различной сложности. Основным преимуществом предложенного подхода является его независимость от структуры входных данных, что позволяет обрабатывать как простые табличные данные, так и сложные вложенные форматы. Это делает инструмент универсальным и применимым к широкому кругу задач системного анализа.

Для реализации пользовательского интерфейса использовался ReactJS, а логика обработки и хранения данных была построена на основе Python и ElasticSearch. Поддержка различных типов данных, возможность динамического изменения графа и наличие интерактивных механизмов фильтрации обеспечивают аналитикам широкие возможности контроля над процессом анализа. Это позволяет эффективно использовать систему как опытным специалистам, так и пользователям с минимальной технической подготовкой.

Таким образом, предложенный метод и инструмент построения графовых моделей представляют собой мощное средство для выявления как явных, так и неявных связей между объектами, независимо от структуры и типа данных. Они могут быть использованы в различных областях: от научного анализа и библиометрии до политического и экономического мониторинга, обеспечивая аналитикам качественно новый уровень понимания сложных систем и взаимосвязей внутри них.

Выводы и заключение

В современных условиях, когда объемы научно-технических данных стремительно растут, важными становятся инструменты, способные преобразовать исходные массивы в наглядные и удобные для анализа графовые структуры. В рамках данного исследования разработан метод построения графа на входных данных различной структуры и выявления явной и неявной связи между объектами, а также инструмент графовой визуализации, который позволяет решить проблему обнаружения взаимосвязей между информационными объектами с помощью механизмов выбора узлов и способа их связи по интересующим данным, настройкой графа и его визуального отображения.

Результаты апробации предложенного метода и инструмента позволяет анализировать два прикладных сценария, связанных с построением отношений между объектами, на примере выявления тематик публикаций по странам и организациям: для этого использованы два набора данных: 5 280 документов по тематике ядерных материалов, 36 008 научных публикаций ОИЯИ.

Список литературы

1. Козицын А. С. Нахождение скрытых зависимостей между объектами на основе анализа больших массивов библиографических данных // VI Международная конференция «Актуальные проблемы системной и программной инженерии» (АПСПИ 2019). Москва, 2019.
2. Тимонин А. Ю., Бождай А. С. Методы анализа гетерогенных данных для построения социального профиля // Russian journal of management. 2017. Т. 5, № 3. С. 481-489.
3. Lindley D. V. Regression and Correlation Analysis. London: Palgrave Macmillan UK, 1990.
4. Georgi M. Methods for Mining Causality from Observations in Artificial Intelligence // Izvestiya SFedU. Engineering Sciences. 2023. Vol. 3, no. 192. Pp. 125-134.
5. Батищев С. В., Искварина Т. В., Скобелев П. О. Методы и средства построения онтологий для интеллектуализации сети интернет // Известия Самарского научного центра РАН. 2002. Т. 4, № 1. С. 91-103.
6. Батура Т. В. Методы и системы семантического анализа текстов // Программные продукты и системы: Международный журнал. 2016. Т. 12.