МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего профессионального образования

**Национальный исследовательский ядерный университет «МИФИ»**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Факультет кибернетики и информационной безопасности

Кафедра системного анализа

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К УЧЕБНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ**

**И КУРСОВОМУ ПРОЕКТУ ПО КОРПОРАТИВНЫМ ИНФОРМАЦИОННЫМ СИСТЕМАМ**

**НА ТЕМУ:**

**Визуализация онтологического представления документальной информации**

|  |
| --- |
|  |
|  |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Студент*  (**Максимов Н.В.** | **Бондаренко Юрий Александрович** | | | | (**Бондаренко Ю.А.**) |
| *Руководитель*  (Фамилия И.О.)  (подпись) | | | (подпись) | | ( ) |
| *Консультант*  (Фамилия И.О.)  (подпись) | | |  | | ( ) |
| *Оценка* | |  | | | |
| *Подписи членов экзаменационной комиссии* | | | |  | |

Москва – 2016 г.

//TODO:

//объем страниц(с. 1)

//нумерация с 3его листа(отображение)

//схема БД

//можно написать про json

**Аннотация**

Отчет 85 с., 2 ч., 4 рис., 2 табл., 3 источника, 2 прил.

Ключевые слова: визуализация, знания, онтологии, операции над онтологиями, данные, объединение, пересечение, масштабирование, проекция

Объектом исследования является онтологическое представление документальной информации.

Цели работы: исследовать особенности визуализации знаний (переход математика => реализация) и создать программную систему, визуализирующую онтологический подход к идентификации информации с помощью мультиграфов.

Критерии оценки системы: системе дается положительная оценка, если пользователь, использующий её, имеет возможность получить доступ к информации о том или ином термине или понятии.

Во введении обосновывается актуальность выбранной темы, формулируются цель и задачи исследования, указывается объект и предмет исследования.  
Первая глава посвящена исследованию теоретических вопросов, в ней раскрываются понятие онтологий и операций над ними.

Во второй главе приводится сравнительная характеристика графических библиотек по определенным критериям.

В третьей главе описывается реализация операций над онтологическим представлением информации.

Заключение содержит основные выводы и результаты разработки.

**Содержание**

1. Список используемых сокращений…………………………………………………..4
2. Введение………………………………………………………………………………..5
   1. Способы представления знаний………………………………………………….5?
3. Основная часть работы………………………………………………………….……..6
   1. Определение онтологий……………………………………………………….….6
   2. Операции над онтологиями………………………………………………………8
      1. Операция объединения……………………………………………………9
      2. Операция пересечения……………………………………………………10
      3. Операция проекции (построение аспектного представления)………....10
      4. Операция масштабирования……………………………………...………10
   3. Выбор графической библиотеки…………………………………………...……11
   4. Структура базы данных
   5. Реализация операций над онтологиями…………………………………………
      1. Операция объединения
      2. Операция пересечения
      3. Операция проекции
4. Заключение…………………………………………………………………………..
5. Список используемых источников……………………………………………………..
6. Приложения………………………………………………………………………………
7. **Список используемых сокращений**

ПОЗ – поисковой образ запроса

ПОД – поисковой образ документа

ПрО – предметная область

ИПС – информационно-поисковая система

JS – язык программирования JavaScript

1. **Введение**

Будем считать, что каждый из нас может иметь свои представления в некоторой области знания. И перед каждым из нас рано или поздно возникает задача предъявить эти представления как самому себе, так и другим людям. Одним из современных направлений решения этой задачи является графическое изображение структуры этих представлений с помощью компьютера.

Понятие информации и информационного поиска всегда, так или иначе, связывается с процессом, имеющим неопределенность исхода, и, если это управляемый процесс, – с выбором, который, в свою очередь, использует данные, находящиеся вне ИПС - с наличными знаниями. Неопределенность такого выбора обусловлена последовательными преобразованиями в связываемых посредством ИПС цепочках: «*знания – информация – документ – ПОД*» и «*проблемная ситуация – задача – запрос – ПОЗ*».

Процесс информационного поиска – это построение множества документов, формально соответствующих ПОЗу, посредством процедур, реализующих ту или иную поисковую модель. Здесь необходимо учитывать, что каждое преобразование как в цепочке «знание => ПОД», так и в цепочке «Проблемная ситуация => ПОЗ» представляет собой отображение, причем в пространствах с меньшим разнообразием. Более того, пространства в обеих цепочках для каждого преобразования хотя и подобны (имеют одинаковую природу), как процесс, сводящийся к отбору через сравнение, в общем случае, гипотетического отыскиваемого объекта с объектами, хранящимися в массиве, реализуется не через сравнение самих объектов, а через соотнесение их хорошо структурированных формализованных описаний – поисковых образов.

Следует отметить еще одно характерное отличие в формировании и использовании образов в машинной среде ИПС и в сознании человека. Машинные образы создаются обычно в виде статичного набора атрибутов (устойчивой структуры) для отражения *наиболее* *характерных* свойств. В сознании образы формируются преимущественно вследствие действий и практически не существуют вне связей. Соответственно, машинный отбор образов реализуется по точным критериям, соотносящим исключительно значения (величины) признаков. Поиск же образов в сознании человека производится по ассоциациям (связям), обычно по признаку целевого (предполагаемого) *использования* значения.

В этом смысле поисковые образы, построенные на предложенных онтологических подходах, позволяют работать равно как с признаками, определяющими свойства, так и с признаками, определяющими их взаимосвязь (поведение).

* 1. **Определение онтологий**

Рассматривая онтологию ПрО с узкой точки зрения идентификации содержания документа при информационном поиске, будем представлять её как совокупность трех систем – функциональной, понятийной и терминологической.

Функциональная система («рабочий интерфейс» онтологии в деятельности субъекта) представляет объекты ПрО и отношения между ними средствами знакового уровня.

Эти отношения имеют функциональную окраску, т.к. определяют существенные с точки

зрения задач пользователя способы и характер совместного существования и использования объектов.

Однако функциональная система не может рассматриваться вне связи со сложившейся в ПрО на *текущий момент времени* системой понятий, которая обычно фиксируется в форме тезаурусов, рубрикаторов, классификационных схем (логико-семантический

базис онтологии).

В понятийной системе объектами (представленными также средствами знакового уровня) являются устойчивые понятия ПрО, а набор отношений ограничен родовидовыми и ассоциативными.

Терминологическая система в онтологии отражает свойства ЕЯ на уровне знаков -терминов ПрО, которые могут быть связаны отношениями эквивалентности (синонимии)

и включения (образования словосочетаний). В качестве термина выступает отдельное слово или словосочетание ЕЯ (а также шифр классификации), которое может быть использовано для описания понятия или объекта в рамках заданной ПрО.

Для определения (установления) взаимосвязи этих систем (а также и их влияния

друг на друга в процессе развития) может быть положена простая операция тождества

элементов на знаковом уровне.

Таким образом, некоторая онтология ПрО (с точки зрения задач информационного

поиска) формально может быть определена, как:

*O =<* , , , ≡ *>*, где

*-* функциональная система;

*-* понятийная система;

*-* терминологическая система;

≡ - операция сопоставления элементов различных систем на уровне знаков, обеспечивающая их тождество в функциональной, понятийной и терминологической системах.

Определенная таким образом онтология также обладает свойствами системы (с т.з.

ОТС), т.е. для отдельной предметной области (в том числе и отдельного её элемента) может быть построено столько онтологий, сколько может быть аспектов рассмотрения ПрО,

т.е. . Однако с целью упрощения выкладок, если не будет явной необходимости, мы будем опускать соответствующий индекс.

Отметим, что в качестве функционального (и терминологического) компонента

может быть и система понятий ПрО. В этом случае её понятийной системой будет система

метапонятий (это отвечает, например, определению «метаонтологии» [Nirenburg], где на

верхнем уровне есть лишь три категории: «объект», «процесс» и «роль»).

Рассмотрим компоненты, входящие в приведенное выше определение онтологии.

Функциональная система определяется как:

*=<* , , , *>,* где

- множество знаковых описаний объектов ПрО;

- множество характеристических атрибутов;

– множество функциональных отношений;

– закон композиции, в соответствии с которым выбрано конкретное системное

основание {, , }

Со структурной точки зрения функциональная система может быть представлена

помеченным взвешенным направленным мультиграфом = <, >,

где - множество вершин, ;

- множество дуг, .

Каждая дуга задается тройкой:

, где - вершина начала дуги; - вершина завершения дуги; - вес дуги (идентификатор функционального отношения).

Направленный мультиграф может иметь дуги и , , но не может при этом содержать дугу

Понятийная система определяется как:

*=<* , , , *>,* где

- множество знаковых описаний понятий ПрО;

- множество характеристических атрибутов знаковых описаний;

– родо-видовые и ассоциативные отношения;  
– закон композиции, в соответствии с которым выбрано конкретное системное основание {, , }.

Со структурной точки зрения понятийная система представлена помеченным взвешенным направленным графом и (с точки зрения введения операций над онтологиями) описана как = <, >, где  
 - множество вершин, ;  
 - множество дуг, .  
Каждая дуга задается тройкой:  
, где- вершина начала дуги; - вершина завершения дуги; - вес дуги (идентификатор отношения).

Направленный граф не содержит симметричных дуг, любые две вершины могут быть соединены только одной дугой.

Терминологическая система:  
*=<* , , , *>*  
- множество терминов ПрО;  
- множество характеристических атрибутов терминов;  
–отношения эквивалентности и включения;  
– закон композиции, в соответствии с которым выбрано конкретное системное основание {, , }.  
Со структурной точки зрения терминологическая система описывается *n*-связным  
графом , где каждая компонента связности представляет собой полный граф (эквивалентность), дерево (включение) или результат операции объединения полных графов и деревьев (при наличии общих вершин).

* 1. **Операции над онтологиями**

Исследования в области онтологических представлений знаний интенсивно, хотя и достаточно эклектично развивающееся направление. Это в значительной степени относится и к введению операций над онтологиями. В многочисленных работах предлагаются достаточно разнообразные множества операций (обзоры достаточно обстоятельно представлены, например, в [FIPA 98, Proceedings, Левашова]), которые достаточно условно можно отнести к классам (1) аналоги теоретико-множественных операций (пересечение, объединение, разность); (2) операции извлечения и удаления фрагментов онтологий для использования при создании новых; (3) операции проверки логической эквивалентности онтологий, перевода онтологий на другой формальный язык и т.п.; (4) операции-функции – средства поддержки онтологий в инструментальных системах (создание, копирование, вывод и т.д.).  
Рассматриваемые далее операции предназначены для построения структурных комбинаций информационных объектов (документов, сообщений), представленных онтологиями, на которых субъект может построить новое знание (задачи вывода нового знания на имеющихся онтологиях не рассматриваются).

В качестве основных операций над онтологиями (на структурном уровне) рассмотрим операции из 1-го и 2-го классов - бинарные операции объединения и пересечения и унарные - построения аспектного представления и масштабирования онтологий, с помощью которых можно, в том числе, синтезировать новые онтологии, отражающие предметную область в аспекте, заданном пользователем.

В силу того, что операции над онтологиями рассматриваются в рамках одной ПрО, будем считать, что исходные онтологии имеют общие понятийную и терминологическую системы. Для формализации операций над онтологиями необходимо определить функцию подобия, вычисляющую меру соответствия при сравнении понятий и связей в исходных онтологиях с целью выделения семантически схожих элементов [Steffen Staab].

Введем функцию подобия *sim*(*a*,*b*), обладающую следующими очевидными свойствами:

1. *Sim*(*a*, *b*) ∈ [0;1] ;

2. *Sim*(*a*, *b*) = 1 → *a* = *b* (элементы *a*, *b* тождественны);

3. *Sim*(*a*, *b*) = 0 → *a* ≠ *b* (элементы *a*, *b* различны);

4. *Sim*(*a*, *a*) = 1 (свойство возвратности);

5. *Sim*(*a*, *b*) = *Sim*(*b*, *a*) (свойство симметричности).

Функция подобия рассчитывается для элементов функциональных систем пары исходных онтологий *O1* и *O2* (в дальнейшем описании индекс *i1* характеризует принадлежность элемента онтологии *O1*, а индекс *i2* - онтологии *O2*) по следующим правилам.

Для вершин мультиграфов:

1. *sim* = 1 , если ≡

2. *sim* = , если в графе общей понятийной системы онтологий существуют принадлежащие одному маршруту вершины и такие, что ≡   
и (*n* – длина маршрута, ( *j* = ) - вес отдельной дуги).

3. *sim*, если в графе общей терминологической системы онтологий  
существует полный подграф, содержащий вершины и такие, что ≡   
и (*p* – количество вершин подграфа)

4. *sim*, если в графе общей терминологической системы онтологий  
существует цепь между вершинами и такими, что ≡   
и (*q* – длина цепи)

5. *sim*, если не применимо ни одно из правил 1-4.  
Для дуг мультиграфов:  
*sim*

Результат бинарных операций над онтологиями , , и  
, , , приведенными к одному понятийному и терминологическому основанию, представляет собой онтологию , , . Таким образом, необходимо формализовать бинарные операции для функциональных систем.

Представление функциональных систем как мультиграфов позволяет свести операции над онтологиями к операциям над мультиграфами и

При выполнении операций над мультиграфами будем считать одинаковыми вершинами те, для которых функция подобия отлична от 0, а одинаковыми дугами – те, для которых функция подобия равна 1.

* + 1. **Операция объединения онтологий .**

Формально алгоритм объединения можно представить следующим образом:

1. Вычисляется множество вершин (для вершин разных мультиграфов, принадлежащих множеству , функция подобия равна 1).

2. Для каждой вершины из множества вершин формируется множество маршрутов к вершинам из множества в понятийном графе . Если для  
вершины множество маршрутов не пусто, для каждого из маршрутов вычисляется функция подобия. Две вершины, для которых функция подобия принимает максимальное значение, считаются далее тождественными (в мультиграфах каждая из вершин замещается  
соответствующим «понятийным маршрутом») и при новом пересечении множеств вершин  
мультиграфов формируют множество с учетом понятийной системы.

3. Рассматриваются множества вершин и . При  
наличии пары вершин (по одной из каждого множества), входящих в одну компоненту  
связности графа терминологической системы, происходит замена каждой из вершин соответствующей компонентой (или ее фрагментом, включающим обе вершины) и при новом  
пересечении множеств вершин мультиграфов формируют множество с учетом терминологической системы.

4. Множество вершин мультиграфа объединения онтологий формируется как результат теоретико-множественных операций:

5. Инцидентность дуг из множеств и сохраняется в мультиграфе объединения онтологий (с учетом слияния дуг, для которых функция подобия равна 1).

При объединении функциональных систем исходных онтологий по такому алгоритму могут возникнуть противоречия двух типов.

1. В результате операции объединения мультиграф онтологии может содержать  
вершины из множеств и , которые являются подграфами понятийного и терминологического графов, дуги и ребра которых описывают отношения понятийной и терминологической систем (что противоречит определению мультиграфа функциональной системы).  
Для разрешения этих противоречий необходима экспертная оценка и замена таких под-  
графов узлами (и, возможно, дугами), соответствующими функциональной системе.

2. Сохранение в мультиграфе объединения онтологий инцидентности дуг из множеств и может привести к наличию противоречивых дуг, инцидентных одной и той же паре вершин. Такая ситуация также исследуется субъектом – экспертом.

**3.2.2. Операция пересечения онтологий .**

Отличие алгоритма пересечения от алгоритма объединения состоит в способе формирования результирующего мультиграфа:

,

а инцидентность сохраняется в мультиграфе пересечения онтологий только для тех дуг из множеств и , для которых функция подобия равна 1.

При пересечении функциональных систем исходных онтологий могут возникнуть противоречия только первого из описанных выше типов.

* + 1. **Операция проекции онтологий (построение аспектного представления).**

Аспект рассмотрения (представления, описания) в свою очередь может быть задан функциональной системой *=<* , , , *>,*  тогда операция проекции может быть сведена к операции пересечения исходной – *O =<* , , , ≡ *>* и аспектной – , , онтологий:  
. В результате операции пересечения онтологий на самом деле происходит «обогащение» аспектного описания с учетом понятийной и терминологической систем.

Рассмотрим следующие возможные ситуации.

1**.**  - аспект задается на уровне объектов – знаковыми описаниями совокупности объектов. Мультиграф аспектной онтологии, участвующий в операции пересечения, при этом представляет собой пустой граф . В этом случае для множества вершин результирующего мультиграфа операция пересечения выполняется по правилам, описанным выше, а множество дуг формируется из дуг множества , инцидентных вершинам множества .

2. - аспект задается на функциональном уровне – множеством  
функциональных отношений. В этом случае множество дуг мультиграфа проекции  
 , а множество вершин формируется из вершин, инцидентных дугам из  
множества .

3. - аспект задается на смешанном, объектно-функциональном  
уровне, и для операции пересечения может быть сформирован мультиграф  
 с непустыми множествами вершин и дуг.

**3.2.4. Масштабирование онтологий.**

Для описания операций масштабирования (укрупнения или детализации) онтологии определим для исходной онтологии *O =<* , , , ≡ *>*  
«онтологию масштабирования» как , , , где и сведем операции масштабирования к разрешению противоречий первого типа в онтологии  
в соответствии со следующими правилами:

- в случае укрупнения исходной онтологии: в узлах, содержащих дерево понятий,  
остаются понятия самого верхнего уровня, а в узлах, содержащих подграфы терминологической системы – отдельные слова;

- в случае детализации исходной онтологии: в узлах, содержащих дерево понятий,  
остаются понятия самого нижнего уровня, а в узлах, содержащих подграфы терминологической системы – словосочетания.

* 1. **Выбор графической библиотеки**

Перед тем, как начать реализовывать визуализация онтологического подхода к идентификации информации с помощью мультиграфов, необходимо было провести сравнительный анализ графических библиотек, которые поддерживает JS.

Важными свойствами являются:

1. Возможность реализации визуализации мультиграфа
2. Простота использования библиотеки
3. Быстродействие
4. Функционал
5. Качество документации

Рассматривались следующие графические библиотеки:

1. D3.js
2. Arbor.js
3. CytoScape.js

При сравнении библиотек необходимо ввести понятие веса критерия сравнения. Под весом критерия будем подразумевать его значимость (важность) для реализации текущей задачи. Таким образом, самыми значимыми будут критерии «Быстродействие» и «Функционал».

Рассмотрим эти два критерия.

1. Функционал
   1. CytoScape.js

+ присутствует возможность визуализации графов;

+ присутствует возможность добавления надписей к вершинам графа;

- отсутствует возможность добавления надписей к ребрам графа;

- отсутствует разнообразие стилей отрисовки вершин и ребер графа (необходимо, чтобы реализовать наиболее «приятный» интерфейс для пользователя).

1.2. Arbor.js

+ присутствует возможность визуализации графов;

+ присутствует возможность добавления надписей к вершинам графа;

+ присутствует разнообразие стилей отрисовки вершин и ребер графа

- отсутствует возможность добавления надписей к ребрам графа;

1.3. D3.js

+ присутствует возможность визуализации графов;

+ присутствует возможность добавления надписей к вершинам графа;

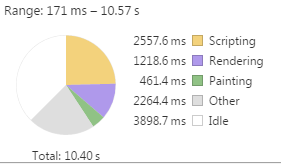
+ присутствует разнообразие стилей отрисовки вершин и ребер графа

+ присутствует возможность добавления надписей к ребрам графа;

Наибольшую оценку по критерию «Функционал» получает библиотека D3.js

1. Быстродействие

Был проведен сравнительный анализ скорости отрисовки графов для всех трех библиотек на малопроизводительном компьютере. Результаты: все библиотеки работают без нареканий, но при сравнении быстродействия, самой быстрой оказалась библиотека D3.js

D3.js

Ниже приведена сравнительная таблица характеристик графических библиотек. Все оценки наименее весомых критериев (все, кроме основных: «функционал» и «быстродействие») были поставлены на основании субъективного мнения разработчика.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерии | Вес критерия | Название оцениваемой библиотеки | | |
| D3.js | Arbor.js | CytoScape.js |
| Оценка | | |
| Простота использования библиотеки | 4 | 7 | 9 | 5 |
| Быстродействие | 10 | 10 | 8 | 9 |
| Функционал | 10 | 10 | 6 | 5 |
| Качество документации | 8 | 8 | 7 | 5 |
| Итого | |  |  |  |

Установка всех трех библиотек проходила идентичным образом: либо указанием прямой ссылки на файл в интернете *(<script src=https://d3js.org/d3.v3.min.js ></script>*), либо скачиванием необходимых файлов с официального сайта и указанием ссылки на него (*<script src="js/arbor.js"></script>* ).

Arbor.js является удобной библиотекой для работы с графами, легким к пониманию и широким набором функций для работы с ними. Простейший пример можно сделать буквально за пару минут. Однако, если посмотреть на функционал этой библиотеки, то выяснится, что он недостаточен для полноценной работы с графами

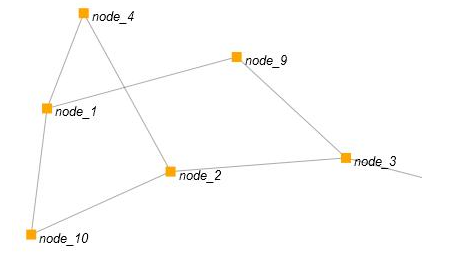
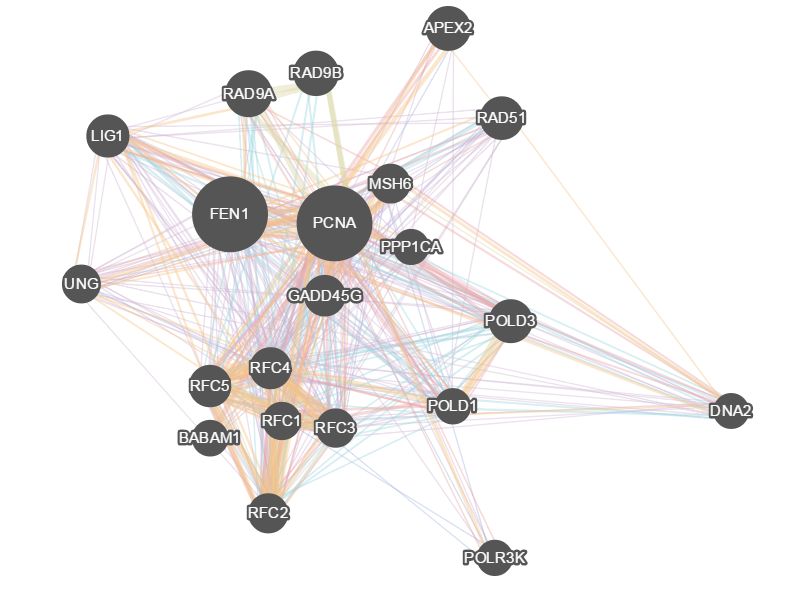
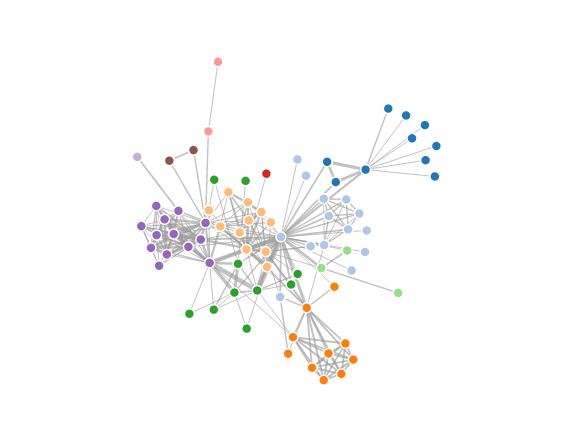


Рисунок .Простейший пример использования Arbor.js

Cytoscape.js также является библиотекой для работы с графами, однако эта библиотека слишком сложна в освоении и имеет функционал, недостаточный для текущей задачи.



*Рисунок 2. Пример использования Cytoscape.js с сайта разработчика*



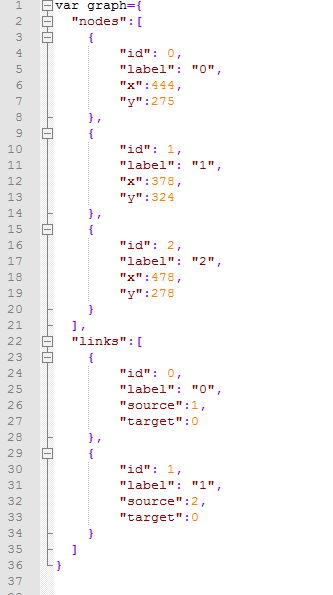
*Рисунок 3. Пример использования библиотеки D3.js*

При сравнении c D3.js, Arbor.js и Cytoscape.js имеют более узконаправленный функционал т.к. эти библиотеки были созданы для работы только с графами, в то время как D3 призван для обработки и визуализации данных. Функций в D3 более чем достаточно: работа с 3D графикой, работа с графами, построение таблиц, различных динамических диаграмм, деревьев и даже карт. Высокая скорость работы в D3 достигается модульностью своих частей, т.е. при желании нарисовать простой кружочек, вы не будете подгружать также функции для рисовки 3D моделей и кривых Безье. Выбран был D3.JS ввиду максимальной широты функционала.

* 1. **Структура базы данных**

База данных представляет собой файл формата json, в котором определены графы. Граф определен как набор массивов его ребер и вершин. Структура каждой вершины графа представляет собой набор параметров, таких как: id, текст на вершине и её координаты (x, y). Структура каждого ребра графа представляет собой набор параметров, таких как: id; текст на ребре графа; id вершины, из которого исходит ребро; id вершины, в которое входит это ребро.

Пример определения одного из графов в программной системе:



* 1. **Реализация операций над онтологиями**
     1. **Операция объединения**

Ниже приведен программный код, реализующий операцию объединения двух графов:

function union(graph1, graph2) {

var nodes = graph1.nodes.concat(graph2.nodes);

var links = graph1.links.concat(graph2.links);

var outgraph = {

nodes: uniqueNodes(nodes),

links: uniqueLinks(links)}

return (outgraph);

}

function uniqueNodes(nodes) {

var result = [];

nextInput:

for (var i = 0; i < nodes.length; i++) {

var str = nodes[i].label;

for (var j = 0; j < result.length; j++) {

if (result[j].label == str) continue nextInput;

}

result.push(nodes[i]);

}

return result;

}

function uniqueLinks(links) {

var result = [];

nextInput:

for (var i = 0; i < links.length; i++) {

var str1 = links[i].source;

var str2 = links[i].target;

for (var j = 0; j < result.length; j++) {

if ((result[j].source == str1)&(result[j].target == str2)) continue nextInput;

}

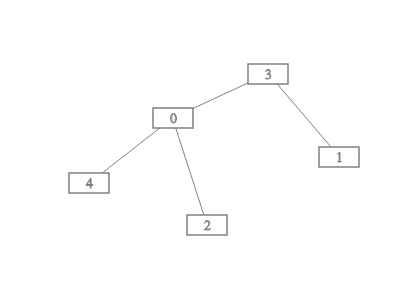
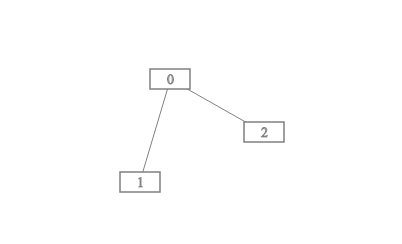
result.push(links[i]);

}

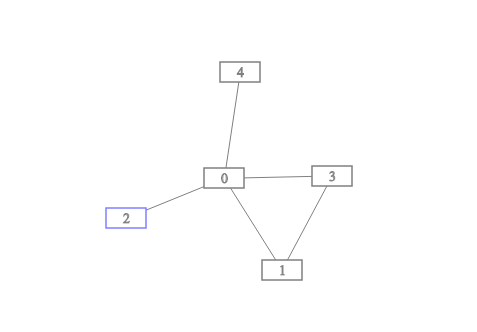
return result;

}

Исходные графы:



Результирующий граф:



* + 1. **Операция пересечения.**

Ниже приведен программный код, реализующий операцию пересечения двух графов:

function intersect(graph1, graph2) {

var outnodes = [];

var outlinks = [];

for (i = 0; i < graph1.nodes.length; i++)

for (j = 0; j < graph2.nodes.length; j++)

if (graph1.nodes[i].label == graph2.nodes[j].label)

{

outnodes.push(graph1.nodes[i]);

break;

}

for (i = 0; i < graph1.links.length; i++)

for (j = 0; j < graph2.links.length; j++)

if ((graph1.links[i].source == graph2.links[j].source)&(graph1.links[i].target == graph2.links[j].target))

{

outlinks.push(graph1.links[i]);

}

var outgraph = {

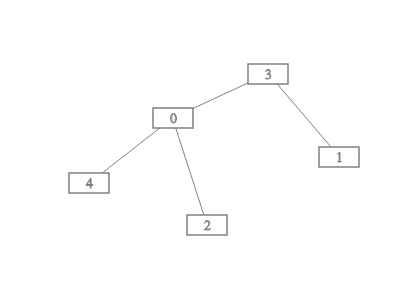
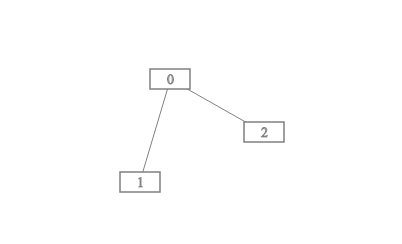
nodes: outnodes,

links: outlinks}

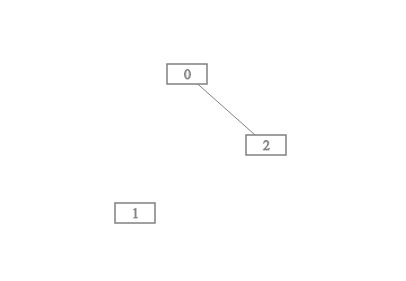
return (outgraph);

}

Исходные графы:



Результирующий граф:



* + 1. **Операция проекции.**

**//пока не работает**

* 1. **Заключение.**

В настоящей учебно-исследовательской работе был рассмотрен онтологический способ представления документальной информации.

Было определено понятие онтологий и четырех операций над ними:

1. Объединение
2. Пересечение
3. Масштабирование
4. Проекция

Был выбран язык программирования, графическая библиотека, структура базы данных.

Также были реализованы три из четырех операций над онтологиями.

В результате была создана программная система, визуализирующая данные в виде взвешенного мультиграфа с возможностью выделения отдельной вершины, выделения нескольких вершин, объединения и пересечения нескольких графов

Приложение.

Внешний вид программной системы, реализованной в итоге:

