предикаты математической логики. Разработки рекурсивных сетей и первые модели появились в середине 1990-х.

Рекурсивные нейронные сети работают с векторными представлениями слов. Векторное представление слов – класс методов и подходов для обработки естественного языка, суть которых состоит в том, что слову словаря ставится в соответствие n-мерный вектор. Для получения векторных представлений используются нейронные сети, которые после обучения могут автоматически выполнять указанную задачу. Векторные представления позволяют программе работать со значениями слов, т.к. смысл векторных представлений состоит в том, чтобы для слов со схожими значениями получались близкие вектора, а для далеких понятий расстояние должно быть большое. На практике используются достаточно большие размерности, от 200 до 500. Таким образом для огромного количества слов из словаря можно получить векторы, которые могут обрабатываться на ЭВМ. С увеличением размерности результат анализа становится точней, а скорость работы и обучения замедляется.

Список использованных источников:
1. Заенцев, И. В. Нейронные сети: основные модели/ И. В. Заенцев. – Воронеж, 1999. – 76 с.
2. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank – Stanford University, 2013.
3. Википедия:Рекурсивные нейронные сети. [Электронный ресурс]. – Режим доступа:https://ru.wikipedia.org/wiki/Рекурсивные_нейронные_сети . – Дата доступа: 25.03.2018.
4. Википедия: Анализ тональности текста. [Электронный ресурс]. – Режим доступа:https://ru.wikipedia.org/wiki/Анализ_тональности_текста . – Дата доступа: 25.03.2017.

SQRT-ДЕРЕВО КАК НЕЗАСЛУЖЕННО ЗАБЫТАЯ СТРУКТУРА ДАННЫХ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Вишневский В.А.

Гербик А.И. – ассистент

В рамках данной работы будет освещена структура данных под названием sqrt-дерево и её преимущество перед другими открытыми структурами данных.

Для объяснения принципа работы sqrt-дерева будет проще рассмотреть следующую теоретическую задачу: “Дан набор из N чисел. На некоторых его отрезках требуется посчитать произведение чисел по заранее заданному модулю M”.

Описание sqrt-дерева лучше начать с идеи разбиения массива на блоки длины $\sqrt{n}$ (здесь и далее под $\sqrt{n}$ имеется ввиду $\lfloor\sqrt{n}\rfloor$), за исключением последнего блока, который может иметь размер меньше, чем $\sqrt{n}$. Принцип разбиения указан на рисунке 1:

$$\left\{a_1,\ldots,a_{\sqrt{N}}\right\},\left\{a_{\sqrt{N}+1},\ldots,a_{2\sqrt{N}}\right\},\ldots,\left\{a_{\left\lfloor\frac{N}{\sqrt{N}}\right\rfloor\cdot\sqrt{N}+1},\ldots,a_N\right\}$$

Рис. 1 – принцип разбиения на блоки

Предпросчитаем для каждого блока: произведение всех чисел, принадлежащих ему, произведение чисел на каждом префиксе и суффиксе этого блока. А также для каждого отрезка блоков посчитаем произведение чисел. В итоге суммарное время работы предпросчета будет $O(n)$. Используя эти данные, мы уже можем решать вышеприведенную задачу за $O(\sqrt{n})$ времени на запрос. Заметим, что мы можем отвечать за $O(1)$ на все запросы, кроме тех, которые затрагивают элементы ровно одного блока. Чтобы эффективно отвечать на такой тип запросов, мы можем построить в этом блоке такую же структуру данных, как была описана выше. Будем выполнять это действие рекурсивно для каждого блока, который имеет размер больше единицы. Таким образом мы имеем древообразную структуру, изображенную на рисунке 2:

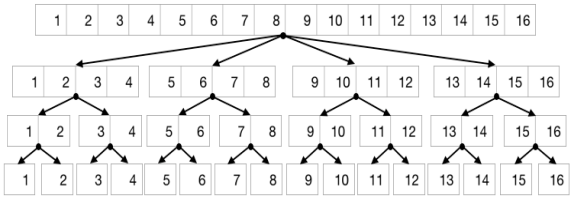

Рис. 2 – sqrt-дерево

На каждом слое находится ровно N вершин, поэтому для оценки сложности работы нам достаточно найти количество слоёв. Давайте заметим, что новый слой создается только когда предыдущий слой имеет

размер больше одного, а значит нам нужно найти количество взятий операции корня к числу N для того, чтобы оно обратилось в единицу. Количество таких операций равно $O(\log\log n)$. Теперь у нас есть всё, чтобы оценить необходимые ресурсы для работы структуры данных:

- Время работы предпроцесса: $O(n\log\log n)$.
- Необходимая память: $O(n\log\log n)$.
- Время ответа на один запрос: $O(\log\log n)$ или $O(1)$, при использовании битовых операций для определения нужного слоя.
- Время модификации элемента: $O(\sqrt{n})$.

В данной структуре без проблем можно использовать любую операцию, для которой выполняется свойство ассоциативности. Для примера, такими операциями являются: сложение, умножение, а также все битовые операции.

Немного о скорости относительно других известных структур данных на бенчмарке состоящем из 10^7 запросов и 10^5 элементов.

алгоритм\ количество модификаций	0	100	10^5	10^7
Segment tree	10.11 сек.	10.28 сек.	11.78 сек.	12.54сек.
Sqrt tree	2.01 сек.	2.28сек.	4.92сек.	13.28сек.
Sqrt decomposition	138.17 сек.	138.75сек.	140.32сек.	162.83сек.

Как видно структура работает в несколько раз быстрее, чем другие структуры данных, пока дело не доходит до большого числа изменений.

Список использованных источников:
1. N Alon, B Schieber, Optimal preprocessing for answering on-line product queries
2. Т. Кормен, Алгоритмы. Построение и анализ
3. Sqrt Decomposition, e-maxx.ru

ГРАФОВЫЕ БАЗЫ ДАННЫХ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Чурин А.П., Владыко В.Д.

Теслюк В.Н. – к.ф-м.н., доцент

В настоящее время есть потребность для обработки разноордных данных, которые имеют непредсказуемую структуру. Которая может превратиться либо в BigData, либо в сложную семантическую сеть, и зачастую мы не можем предвидеть, какой она будет.

Как альтернатива базам данных SQL с 2000-х годов развивается направление NoSQL. В эту категорию попадают – от иерархических и сетевых БД до упрощённых БД, которые имеют подобие хэш-таблиц, и документарные БД. Причиной эволюции базы данных заключается в следующем: если программный продукт оперирует однотипными и примитивными данными, то можно использовать классический подход к разработке БД. Но если нужно обратиться к 10, 100, 1000 таблицам, чтобы получить данные, то реляционная база данных начинает работать медленно, а написание такого запроса займет довольно много времени.

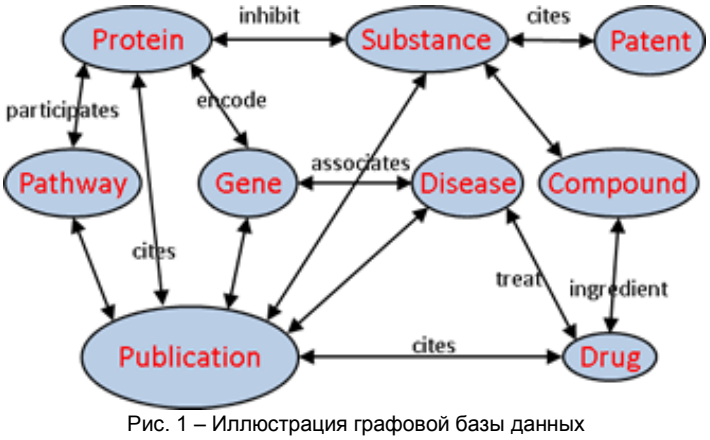

Рис. 1 – Иллюстрация графовой базы данных