САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет

Кафедра системного программирования

Перевалова Марина Андреевна

Алгоритм приближенного выполнения операции соединения по подобию

Дипломная работа

Допущена к защите.

Зав. кафедрой:

д.ф.-м.н., проф. Терехов А.Н.

Научный руководитель:

д.ф.-м.н., проф. Новиков Б.А.

Рецензент:

доцент Графеева Н.Г.

Санкт-Петербург

2013

SAINT-PETERSBURG STATE UNIVERSITY

Mathematics&Mechanics Faculty

Software Engineering Chair

Marina Perevalova

Approximate similarity join algorithm

Graduation Thesis

Admitted for defence.

Head of the chair:

Professor A.N. Terekhov.

Scientific supervisor:

Professor B.A. Novikov.

Reviewer:

Associate Professor N.G. Grafeeva.

Saint-Petersburg

2013

Оглавление

1. Введение

Быстрый рост объемов и разнообразности информации приводит к усложнению задачи информационного поиска. Все чаще возникает потребность обработки не только структурированных, но и сложных объектов, таких как тексты, изображения и графы. Фундаментальное их отличие состоит в том, что сложные объекты могут иметь атрибуты, нечеткие по своему существу. Так мы не можем сказать, что определенное изображение «зеленое» или «не зеленое», но мы можем оценить уровень цвета в пределах промежутка [0, 1], где 0 означает совсем не зеленое, а 1 – абсолютно зеленое.

Соединение данных

С ростом количества информации, растет и количество поисковых систем, обрабатывающих запросы, рассчитанные на разные типы данных. В связи с этим всегда было ценным умение сравнивать и соединять информацию из различных источников.

Гетерогенность обрабатываемых данных и их слабая структурированность приводит к возникновению сложных запросов, требующих комбинирования различных техник соединения информации.

Приведем несколько примеров запросов такого рода:

- Сбор информации, полученной из разных источников, об одном объекте;
- Агрегирование множественных оценок одного объекта в единую оценку.

При работе со сложными запросами, операции соединения с использованием предиката бывает недостаточно из-за нехватки точности. Мы не можем однозначно сказать, связан или нет какой-то объект с запросом или другим объектом, нам нужен более гибкий критерий, позволяющий сравнивать объекты друг с другом. В качестве такого критерия в данной работе выступает понятие подобие.

Подобие оценивает степень похожести объектов и используется при решении многочисленных задач, работающих с информацией. Количественным способом определения подобия являются функции подобия, оценивающие близость объектов по шкале [0, 1].

Данная работа посвящена операции соединения по подобию (similarity join), основывающейся на понятии подобия для соединения данных. Соединение по подобию нашло множество применений в таких областях, как очистка данных, интеграция

данных и data mining. Операция соединения в традиционных базах данных так же может быть рассмотрена, как форма соединения по подобию с дискретной функцией подобия, возвращающей 0 или 1.

Данная работа выполнена в контексте обработки сложных запросов. Как уже было отмечено, обработка таких запросов требует комбинации соединения информации с поиском по подобию и учета других атрибутов объектов. В связи с этим, основной целью работы являлась разработка алгоритма соединения, принимающего во внимание не только подобие, но и другие оценки объектов.

Традиционная форма операции соединения по подобию использует заданный порог и заданную функцию подобия для нахождения всех пар, чье подобие выше данного порога. Однако это требует ввода конкретного порога подобия, а в большинстве случаев порог не известен заранее и может сильно колебаться в зависимости от разных наборов данных.

Привлекательной альтернативой является нахождение определенного количества наиболее похожих пар. Плюс этого подхода заключается в том, что он не требует подбора оптимального порога вручную, что чаще всего приводит к нескольким неудачным попыткам с пустыми или чрезмерно большими наборами результатов.

Приближенное решение

При больших объемах информации, обработать все имеющиеся данные не всегда представляется возможным. Однако учитывая нечеткость самого понятия подобия, точность результатов порой бывает не таким принципиальным критерием. Точное выполнение запроса, приблизительного по своей натуре, не имеет особого смысла, и в ряде случаев, когда временные ресурсы ограничены, больший интерес представляет поиск приближенного результата, допускающего некоторое количество ошибок – потерянных или ложных пар.

Приближенное выполнение задачи открывает много возможностей для оптимизации запросов. При ослаблении требований к качеству результата, уменьшается и время, затрачиваемое на операцию. Маневрирование между быстрым и точным результатом и позволяет оптимизаторам спланировать и оптимизировать запрос.

Суть оптимизации в традиционных базах данных заключается в поиске минимума функции стоимости операции. В случае приближенного решения, необходимо так же учитывать и дополнительный критерий – качество результата. Для возможности

осмысленной оптимизации зависимость ожидаемого качества от времени работы должна быть известна заранее.

Целью данной работы являлась не только разработка алгоритма соединения по подобию с возможностью управления соотношением время/качество, но и постройка модели стоимости, отражающей зависимость этих параметров друг от друга. Задачей алгоритма является отработать за указанное время с ожидаемым результатом, выбор же между скоростью и точностью остается вопросом оптимизации.

2. Обзор существующих разработок в данной области

Идея использования приближенного выполнения задач в целях оптимизации запросов представлена в статье [3]. В ней представлена алгебраическая модель обработки сложных запросов для гетерогенных данных. Данная работа выполнена в контексте этой работы и является реализацией идеи для конкретной операции соединения по подобию.

2.1. Соединение по подобию

Алгоритмическая сложность задачи соединения по подобию заключается в поиске эффективных решений. Наивный алгоритм, вычисляющий подобие всевозможных пар, работает с вычислительной сложностью $O(m \cdot n \cdot cost_{sim})$, где m и n – размеры входных наборов данных, а $cost_{sim}$– стоимость вычисления подобия пары. Однако вычисление подобия является дорогой операцией, и в связи с такими высокими затратами появляется необходимость поиска других подходов.

Среди существующих работ на эту тему можно выделить две группы. Первая посвящена приближенному решению проблемы [8], [11], [16], сокращающему вычислительные затраты за счет сокращения качества результата. Вторая же группа это работы, занимающиеся нахождением точного ответа, то есть всех пар, удовлетворяющих некоторому условию на значение подобия.

Одним из наиболее известных методов приближенного поиска по подобию является Local Sensitive Hashing [8]. Основная его идея заключается в хешировании объектов несколькими хеш-функциями так, чтобы подобные объекты с большей вероятностью оказались в одном слоте хеш-таблицы, чем различающиеся. Затем любая пара, находящаяся в одном слоте считается кандидатом на подобие. Алгоритм допускает как наличие ложных результатов, так и некоторую потерю истинных.

В статье [16] рассматривается задача кластеризации синтаксически подобных веб-страниц. Для решения этой задачи вычисляется сигнатура (sketch) каждого документа и при вычислении подобия сравниваются не сами документы, а эти сигнатуры, за счет чего и теряется точность.

Множество работ, посвященных точному соединению по подобию, используют инвертированные индексы и схему фильтрация-проверка [1], [4], [5], [10], [14], [15]. Основная идея данного подхода состоит в использовании эффективного фильтра, позволяющего исключить пары, которые не могут быть подобными, из множества кандидатов. Само подобие вычисляется только для оставшихся пар, что помогает значительно сократить количество вычислений.

Один из основных алгоритмов точного соединения по подобию, алгоритм All-pairs, представлен в работе [14]. При имеющейся коллекции объектов, заранее отсортированных в порядке увеличения их размеров, он находит все пары, подобие которых больше заданного порога. В алгоритме применены распространенные проверки для сокращения набора кандидатов – префиксное фильтрование и фильтрование по размеру, использующиеся так же и в нашей работе.

Авторы [5] предлагают дополнительные методы фильтрования – позиционное и суффиксное. В статье представлены два эффективных алгоритма – ppjoin и ppjoin+, являющихся модификацией алгоритма All-pairs и включающих в себя описанные методы.

Работа [10] отличается тем, что в ней рассматривается префиксное фильтрование с переменной, а не фиксированной длиной префикса. Алгоритм адаптируется, сравнивая оценки сложности при разной длине и выбирая оптимальную для каждого объекта.

В работе [1] фильтрование основано на сгенерированных сигнатурах объектов. Так же в работе предлагается метод фильтрования по размеру, позволяющий отсекать кандидатов не подходящего размера.

С понятием подобие тесно связано понятие расстояние, которое может быть вычислено с помощью функций расстояния. Интуитивно понятно, что между двумя объектами с высоким значением подобия расстояние будет маленьким. Работа [4] посвящена соединению по подобию, использующему для оценки подобия объектов функцию редакционного расстояния (edit distance). Авторы предлагают новые методы фильтрования, анализирующие расположение и содержимое несовпадающих q-граммов объектов. Алгоритмы, представленные в работе, являются комбинацией этих техник и уже известных алгоритмов.

В статье [9] предлагается альтернативная схема для поиска подобных строк, основанная на построении префиксного дерева. По сравнению со схемой фильтрация-

проверка подход может генерировать подобные пары без шага проверки. Предложенные в работе алгоритмы лучше всего работают на наборах данных с небольшой длиной строк.

Работа [6] интересна нам тем, что, в отличие от предыдущих описанных, не использует заранее заданный порог подобия. Вместо этого предложенный алгоритм находит наилучшие k пар объектов с самым высоким подобием, тем самым смешивая в себе задачу соединения по подобию и задачу top-k запросов.

Так же задача нахождения пар наиболее близких объектов была изучена в работе [7], в которой были предложены алгоритмы поиска лучшей пары и k лучших пар в пространственных базах данных.

2.2. Top-k запросы

Важное место среди работ на тему top-k запросов занимает статья [13]. В ней рассматриваются объекты не с одной оценкой, а с наборами атрибутов и оценками для каждого отдельного атрибута. Итоговая же оценка, по которой выбираются наилучшие k объектов, вычисляется с помощью некоторой агрегирующей функции.

В работе представлены эффективные алгоритмы для агрегации - Fagin’s Algorithm (FA), Threshold Algortihm (TA), и несколько их модификаций. В FA оценки объектов сортируются по убыванию для каждого атрибута, а затем считываются параллельно для всех отсортированных списков. Алгоритм останавливается, когда k объектов просмотрено целиком, т.е. для k объектов были считаны оценки всех атрибутов. Затем уже происходит считывание недостающих оценок для просмотренных в ходе объектов, вычисляются итоговые оценки и выбираются наилучшие k объектов. Стоит отметить, что при считывании недостающих оценок происходит рандомный доступ к диску, а не последовательный, как в случае просмотра оценок по порядку.

В TA, в отличие от FA, считывание недостающих оценок происходит сразу, что позволяет тут же вычислять итоговую оценку всех объектов и хранить в буфере только k лучших. Алгоритм останавливается тогда, когда итоговые оценки всех объектов, хранящихся в буфере на момент очередной итерации, становятся больше определенного порогового значение. Порог же, в свою очередь, вычисляется, как минимальная из оценок атрибутов, прочитанных на данной итерации, т.е. оценок в отсортированных списках.

В нашей работе, как и в описанных алгоритмах, используется агрегация оценок и поиск k лучших, однако нас интересуют оценки объектов и их подобие. Принципиальная разница заключается в том, что в нашем случае приходится учитывать стоимость вычисления функции подобия, и разработанный алгоритм нацелен на всевозможное сокращение числа этих вычислений.

Статья [12] посвящена приближенному вычислению top-k запросов с вероятностными гарантиями. Акцентирована она на алгоритме TA-sorted, модификации алгоритма TA, использующем только последовательный доступ к диску. Основная идея подхода состоит в том, чтобы для каждого просмотренного объекта вычислять вероятность, с которой он может принадлежать лучшим k ответам. Если эта вероятность ниже заданного порога, объект не рассматривается дальше.

В работе [2] так же рассматривается приближенное выполнение top-k запросов. Однако в отличие от предыдущей работы, здесь предложена схема, позволяющая добавлять к имеющимся алгоритмам возможность прерывания. Модифицированные алгоритмы могут предоставить текущий результат в любой момент, но точность ответа, конечно, увеличивается с увеличением времени его работы. Так же авторы предлагают различные техники, позволяющие вычислить вероятностные гарантии качества ответа.

3. Постановка задачи

Целью данной работы являлась разработка приближенного алгоритма соединения по подобию для двух множеств объектов с оценками. Алгоритм ищет наилучшие k пар объектов с наивысшими итоговыми оценками, где итоговая оценка складывается из подобия и оценок объектов. Число k, так же как и желаемое время работы алгоритма, являются заданными параметрами. Задачей алгоритма является отработать за любое отведенное ему время, с разными соответствующими гарантиями качества.

Для достижения цели были поставлены следующие задачи:

- Изучить существующие наработки и алгоритмы в данной области;
- Разработать и реализовать собственный алгоритм, удовлетворяющий перечисленным требованиям;
- Построить расширенную модель стоимости, оценивающую время работы алгоритма и предполагаемое качество результата;
- Протестировать точность разработанной модели.

4. Основные понятия

4.1. Соединение по подобию

Задача соединения по подобию состоит в поиске всех пар объектов, подобие которых выше заданного порога. Объекты в данной работе рассматриваются как множества токенов из конечной совокупности $U = \{\omega_1, \omega_2, \dots, \omega_{|U|}\}$. Функция подобия $\mathrm{sim}(\cdot\,,\,\cdot)$ возвращает значение подобия двух объектов в пределах промежутка [0, 1].

Опр. Для двух заданных множеств R и S и порога θ операция соединение по подобию возвращает все пары объектов из каждого множества, чье подобие не меньше θ, т.е.:

$$\{\langle r, s\rangle | \ \mathrm{sim}(r, s) \geq \theta, r \in R, s \in S\}$$

В данной работе рассматривается вариант соединения по подобию, находящий лучшие k ответов, т.е. операция, возвращающая k пар объектов с наивысшим значением подобия из всевозможных пар.

Для наглядности рассмотрим процесс подробнее на задаче поиска подобных документов. Перед вычислением подобия каждый документ необходимо разбить на множество токенов. Токенами в данном случае могут быть слова или q-граммы – подстроки из q последовательных символов. Так как слова могут встречаться в документе больше одного раза, будем рассматривать каждый повтор одного слова в одном объекте, как новый токен. После этого подобие документов может быть вычислено, как функция подобия от множеств их токенов.

Будем обозначать через $|r|$ размер объекта r, равный количеству токенов в этом объекте. Для сравнения двух объектов токены должны быть отсортированы в некотором едином порядке $\mathcal{O}$, заданном на всем множестве токенов U.

При работе с документами распространенным выбором является сортировка токенов в порядке увеличения их частоты документа. Частота документа df токена равна количеству объектов, содержащих этот токен. Такой порядок удобен для префиксного фильтрования, используемого в нашей работе, так как при нем рассматриваются токены, идущие в начале. А совпадение редких слов обеспечивает большую вероятность схожести объектов, нежели совпадение таких часто встречающихся слов, как предлоги.

Выбор функции подобия во многом зависит от конкретной поставленной задачи и выходит за рамки данной работы. Ниже представлены некоторые наиболее известные функции подобия:

- Жаккара: $J(r,s) = \frac{|r \cap s|}{|r \cup s|}$
- Дайса: $D(r,s) = \frac{2 \cdot |r \cap s|}{|r| + |s|}$
- Косинусная: $C(r,s) = \frac{\vec{r} \cdot \vec{s}}{\|\vec{r}\| \cdot \|\vec{s}\|} = \frac{\sum_i r_i s_i}{\sqrt{|r|} \cdot \sqrt{|s|}}$
- Перекрытия (overlap): $O(r,s) = \frac{|r \cap s|}{\min(|r|,|s|)}$

  Для простоты восприятия, в данной работе не будем нормировать функцию подобия перекрытия, и определим ее, как $O(r,s) = |r \cap s|$

Пример 1.

Рассмотрим два текстовых документа D_r и D_s:

$D_r =$ *"yes as soon as possible"*

$D_s =$ *"as soon as possible please"*

Слово	*yes*	*as*	*soon*	*as*$_1$	*possible*	*please*
Токен	*A*	*B*	*C*	*D*	*E*	*F*
Частота док.	*1*	*2*	*2*	*2*	*2*	*1*

С помощью вышеприведенной таблицы, описывающей переход слов в токены, документы могут быть преобразованы в следующие объекты:

$$r = \{A, B, C, D, E\}$$

$$s = \{B, C, D, E, F\}$$

Стоит заметить, что второе "as" было преобразовано в слово "as_1" в обоих документах. Объекты так же могут быть отсортированы в порядке увеличения их частоты документов O_{df} (обозначим [...]).

$$r = [A, B, C, D, E]$$

$$s = [F, B, C, D, E]$$

Тогда $O(r,s) = 4,\ J(r,s) = \frac{4}{6} = 0.67$ *и* $C(r,s) = \frac{4}{\sqrt{5}\cdot\sqrt{5}} = 0.80.$

В данной работе в качестве конкретной функции подобия рассматривается коэффициент Жаккара, как одна из наиболее используемых функций, определяющих подобие множеств. Однако от этой функции подобия можно перейти к другим, как продемонстрировано в работе [5].

4.2. Сокращение набора кандидатов

Наивным решением задачи соединение по подобию является вычисление подобия для каждой пары объектов и сравнение его с заданным порогом, либо, в нашем случае, выбор k пар с наивысшими оценками. Однако учитывая высокую стоимость вычисления подобия объектов, существует множество эффективных алгоритмов, сводящих к минимуму количество вычислений функции подобия. Основным их принципом является ввод дополнительных, более дешевых проверок, позволяющих определить, может пара достичь необходимого значения подобия или нет. Для пар, не удовлетворяющих этим проверкам, подобие не вычисляется.

Из существующих подходов, в данной работе было выбрано префиксное фильтрование, на котором основано немалое количество алгоритмов [5], [10], [14], [15], некоторые из которых показали себя высоко эффективными и способными обрабатывать десятки миллионов записей.

Идея префиксного фильтрования основана на интуитивном заключении, что если два объекта с упорядоченными токенами похожи, некоторые их фрагменты должны пересекаться друг с другом, в то время, как у непохожих объектов пересечение будет недостаточным. Для возможности такой проверки необходимо перейти от значения конкретной функции подобия к количеству общих токенов объектов.

Продемонстрируем, как это можно сделать:

$$J(r,s) \geq t \Leftrightarrow O(r,s) \geq \alpha, \text{где } \alpha = \frac{t}{1+t}(|r| + |s|) \qquad (1)$$

Таким образом, от ограничения типа $sim(r,s) \geq \theta$ мы переходим к ограничению типа $O(r,s) \geq \alpha$, где α своя для каждой конкретной функции. Это позволяет сформулировать принцип префиксного фильтрования более формально в следующей лемме:

Лемма 1. Принцип префиксного фильтрования.

Рассмотрим совокупность токенов U с некоторым глобальным порядком O и два объекта r и s, токены которых отсортированы в этом порядке. Назовем -префиксом объекта r первые p токенов r. Если $O(r,s) \geq \alpha$, то $(|r| - \alpha + 1)$-префикс r и $(|s| - \alpha + 1)$-префикс s должны иметь по меньшей мере один общий токен.

Помимо этого в работе используется принцип фильтрования по размеру, предложенный в статье [1] и основанный на преобразовании ограничений конкретных функций подобия в ограничения на размер объектов.

$$J(r,s) \geq t \Rightarrow t \cdot |r \cup s| \leq |r \cap s|$$

$$t \cdot |r| \leq t \cdot |r \cup s| \leq |r \cap s| \leq |y|$$

Т.е. для выполнения условия $J(r,s) \geq t$ необходимо выполнение условия

$$t \cdot |r| \leq |s| \quad (2)$$

Стоит отметить, что приведенные выше принципы фильтрования являются необходимыми, но не достаточными условиями, чтобы пара удовлетворяла заданному ограничению на подобие.

Пример 2.

Рассмотрим две коллекции объектов с упорядоченными токенами R и S, и порог для функции подобия Жаккара t = 0.8.

$$r_1 = [C, D, F] \qquad s_1 = [A, B, C, D, E]$$

$$r_2 = [B, C, D, E] \qquad s_2 = [G, A, B, F]$$

$$r_3 = [G, A, B, D, E]$$

Пары $\langle r_1, s_1 \rangle$ и $\langle r_1, s_2 \rangle$ не удовлетворяют условию (2) и, таким образом, не рассматриваются дальше в связи с фильтрованием по размеру.

Для остальных пар длина префикса каждого объекта x равняется $|x| - \alpha + 1$, где α вычисляется по формуле в уравнении (1). Принципу префиксного фильтрования удовлетворяют только пары $\langle r_2, s_1 \rangle$ и $\langle r_3, s_2 \rangle$ и, таким образом, список из первоначальных шести кандидатов сократился до двух пар, для которых уже вычисляется функция подобия Жаккара.

5. Разработанный алгоритм

Алгоритм 1.

Вход: R, S – множества объектов с оценками, отсортированные в порядке уменьшения их оценок. Число искомых пар k. Функция подобия sim.

Выход: k пар объектов $\langle r, s, \theta \rangle$: $r \in R, s \in S$ с наивысшими итоговыми оценками θ.

1 $Buff \leftarrow \emptyset$;
2 while < k elements in $Buff$ **for each** $r \in R, s \in S$ **do**
3 $\theta = \min(A_r, A_s) \cdot sim(r, s)$; /**подсчет итоговой оценки**/
4 $Buff \leftarrow Buff \cup \langle r, s, \theta \rangle$;
5 $T = \min_k \theta$;
6 for each $r \in R, s \in S$ **do**
7 **if** $\min(A_r, A_s) < T$ **then** /**условие остановки**/
8 **break;**
9 **if** $|r| \leq \frac{T}{\min(A_r, A_s)} \cdot |s|$ **then**
10 $\alpha = CountAlpha\left(\frac{T}{\min(A_r, A_s)}\right)$;
11 $p_r = |r| - \alpha + 1$;
12 $p_s = |s| - \alpha + 1$;
13 $U_{rs} = 0$;
14 **for each** (i, j) **such**
15 **that** $i \in [1, p_r], j \in [1, p_s]$ **do**
16 **if** $r[i] = s[j]$ **then**
17 $U_{rs} = U_{rs} + 1$;
18 **if** $U_{rs} \neq 0$ then
19 $Verify(r, s, \alpha, U_{rs})$;
20 return $Buff$;

Алгоритм 1 описывает разработанный алгоритм, включающий в себя описанные выше принципы префискного фильтрования и фильтрования по размеру. Алгоритм принимает на вход две коллекции объектов с оценками, каждая из которых отсортирована в порядке убывания оценок, и заданное число k. Оценки нормированы и принадлежат промежутку [0, 1]. Вдобавок токены объектов упорядочены в неком едином порядке, заданном на множестве токенов.

Результатом работы алгоритма являются k пар объектов, каждый из соответствующей коллекции, с наивысшей итоговой оценкой среди всех остальных.

Итоговая оценка пары объектов $\langle r, s \rangle$ складывается из оценки обоих объектов и их подобия, и вычисляется по следующей формуле:

$$\theta = \min(A_r, A_s) \cdot sim(r, s)$$

Здесь A_r и A_s – оценки объектов.

Основная идея алгоритма состоит в заполнении буфера первыми k парами объектов и последующем их уточнении по ходу работы, пока в буфере не окажутся искомые k пар с лучшими оценками. Если же время работы ограничено, алгоритм можно остановить раньше и в любой момент получить текущие приближенные результаты.

Возникает вопрос о порядке перебора пар. Так как мы заранее не располагаем никакими сведениями о подобии объектов, но знаем их оценки, естественным предположением становится перебор пар в порядке уменьшения их оценок. Таким образом, начиная с некоторого момента оценки будут достаточно низкими, что даже с наивысшей возможной степенью подобия, итоговая оценка будет неудовлетворительной. Это позволяет значительно сократить количество обрабатываемых пар.

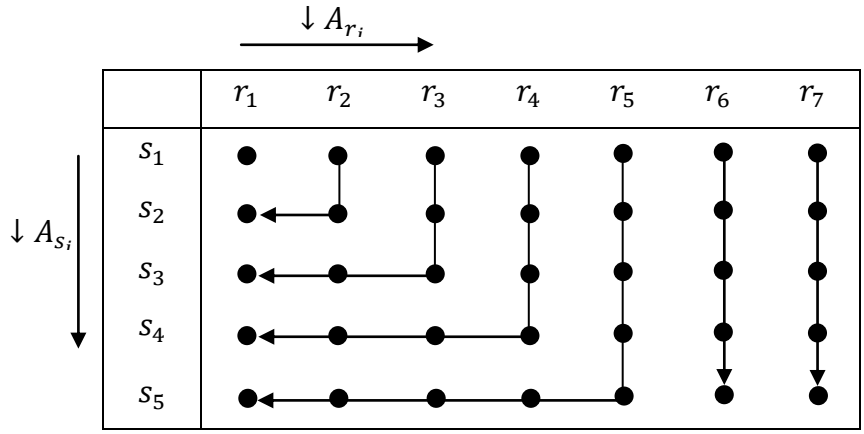

Рис. 1. Порядок перебора пар. Алгоритм обрабатывает пары в порядке, указанном стрелками, начиная с пары $\langle r_1, s_1 \rangle$. Оценки объектов убывают с возрастанием их индекса.

Началом алгоритма служит вычисление итоговой оценки первых k пар объектов и запись их в буфер. Далее в буфере происходит сортировка по итоговым оценкам и текущему порогу T присваивается значение минимальной из них (5 строчка кода).

Порог необходим для того, чтобы понять, в какой момент оценки объектов достигнут значения, дальше которого вести проверку уже бессмысленно.

$0 \leq sim(r,s) \leq 1 \Rightarrow$ если $\min(A_r, A_s) < T$, то и $\theta < T$

А так как оценки объектов только убывают с дальнейшим перебором пар, то итоговые оценки любой последующей пары уже не превысят итоговой оценки минимальной из k текущих пар. Таким образом, при достижении условия остановки $\min(A_r, A_s) < T$ алгоритм завершает свою работу, а текущие k пар в буфере – искомые.

Алгоритм перебирает пары в порядке, изображенном на рис. 1. Для каждой из них объекты проходят проверку сначала на ограничение по размеру (2) (9 строчка), а затем префиксное фильтрование. Заметим, что нас интересуют пары с итоговой оценкой $\theta \geq T$, а значит подобие пары должно удовлетворять условию $sim(r,s) \geq \frac{T}{\min(A_r, A_s)}$.

Чтобы выполнить префиксное фильтрование, функция $CountAlpha\left(\frac{T}{\min(A_r, A_s)}\right)$ преобразовывает это условие в ограничение на необходимое количество общих токенов α, как показано в уравнении (1). Здесь $t = \frac{T}{\min(A_r, A_s)}$.

Алгоритм 2. $\boldsymbol{Verify(r, s, \alpha, U_{rs})}$

Вход: p_r – длина префикса объекта r, p_s – длина префикса объекта s.

1 $\omega_r = r[p_r]$;
2 $\omega_s = s[p_s]$;
3 $O = U_{rs}$;
4 if $\omega_r < \omega_s$ **then**
5 $\quad O = O + |r[(p_r + 1)..|r|] \cap s[(U_{rs} + 1)..|s|]|$;
6 else
7 $\quad O = O + |r[(U_{rs} + 1)..|r|] \cap s[(p_s + 1)..|s|]|$;
8 if $O \geq \alpha$ **then**
9 $\quad \theta = \min(A_r, A_s) \cdot \frac{O}{|r|+|s|-O}$;
10 $\quad Buff = Buff \cup \langle r, s, \theta \rangle$;
11 $\quad T = \min_k \theta$; /* *пересчет порога* */

Для пар, удовлетворяющих префиксному фильтрованию и фильтрованию по размеру, проводится итоговая проверка, заключающаяся в вычислении самого подобия пары, представленного в алгоритме 2. При подсчете количества общих токенов

используется оптимизация, предложенная в работе [5]. Она заключается в том, чтобы сначала сравнить последний токен в префиксах обоих объектов, и искать пересечние только суффикса объекта с меньшим токеном с другим объектом.

Пусть u и v – объекты с последними токенами в префиксах ω_u и ω_v соответственно, и $\omega_u \prec \omega_v$. Префикс u содержит токены, меньшие, чем ω_u, в то время как суффикс v содержит токены, большие, чем ω_v. Тогда благодаря единому порядку, в котором отсортированы токены всех объектов, префикс u не будет пересекаться с суффиксом v. И так как количество общих токенов в префиксах уже подсчитано, остается только найти пересечение суффикса u и объекта v.

Количество вычислений может быть сокращено еще больше, если пропустить первые O токенов объекта v, где O – количество общих токенов в префиксах u и v. Это никак не повлияет на результат из-за того, что как минимум O токенов v пересекаются с префиксом объекта u, а значит не пересекаются с его суффиксом. Описанная оптимизация реализована в 4-7 строках кода алгоритма 2.

Каждая пара, итоговая оценка которой больше текущего порога T добавляется в буфер, а пара с наименьшей итоговой оценкой удаляется из буфера. Так в буфере всегда остается k пар объектов. Алгоритм завершает свою работу при достижении условия остановки или, при ограниченном времени работы, по обрабатыванию определенного числа пар.

Основная сложность состоит в преобразовании отведенного времени в количество пар, которых может успеть обработать за это время алгоритм. Для этого в работе представлена модель стоимости, оценивающая вычислительную сложность и приблизительное время работы алгоритма.

6. Модель стоимости

6.1. Время работы алгоритма

Как было упомянуто выше, приближенная версия алгоритма заканчивает свою работу не по истечению отведенного времени, а по обработке определенного количества пар объектов. Время выполнения алгоритма индивидуально для каждой машины и очевидным образом зависит от мощности имеющихся вычислительных ресурсов. Ниже приведена оценка времени работы алгоритма в зависимости от следующих естественных параметров:

$$time = (k \cdot (5 + 2l_{ob} + logk) + 3c_{all} + c_s \cdot (l_{pr}^2 + 1) + c_p \cdot (2l_{ob} - l_{pr} + 2) + c_v \cdot (2 + k)) \cdot t_{sys}$$

k – желаемое количество пар в результате;

l_{ob} – средний размер объекта;

l_{pr} – средняя длина префикса объекта;

c_{all} – количество пар объектов, обработанных алгоритмом (без учета первых k);

c_s – количество пар, удовлетворяющих фильтрованию по размеру;

c_p – количество пар, удовлетворяющих префиксному фильтрованию;

c_v – количество пар, добавляемых в буфер;

t_{sys} – среднее системное время, затрачиваемое на:

- чтение/запись;
- сравнение;
- одну вычислительную операцию.

Рассмотрим данную формулу более подробно. Первое слагаемое отражает время обработки первых k пар объектов. Сюда включено чтение объектов, вычисление итоговой оценки, запись пары в буфер, а так же сортировку буфера. Токены объектов отсортированы, поэтому время вычисления подобия объектов оценивается как $2l_{ob} \cdot t_{sys}$.

Второе слагаемое – чтение объектов, проверка условия остановки и проверка условия фильтрования по размеру. Третье слагаемое отражает стоимость сверки префиксов объектов, которая производится только для пар, прошедших фильтрование по размеру. Подобным образом вычисление подобия, описанное в алгоритме 2,

проводится только для объектов, удовлетворяющих префиксному фильтрованию, и представлено в четвертом слагаемом. Ну и наконец, последнее слагаемое учитывает запись в буфер пары объектов, чьи итоговые оценки выше текущего порога T, и поиск соответствующего места для вставки.

Таким образом, мы получили зависимость времени работы алгоритма от желаемого количества пар k и некоторых статистических параметров. Чтобы выразить количество обрабатываемых пар c_{all} при заданном времени, необходимо оценить остальные параметры. Каждая проверка отсекает некоторое количество пар, не участвующих в дальнейших вычислениях. Грубую оценку можно получить, сделав предположение о равномерном распределении оценок, и вычислив интересующие нас параметры, как математическое ожидание.

$0 \leq c_{all} \leq |R| \cdot |S| - k \Rightarrow c_{all} \approx \frac{|R| \cdot |S| - k}{2}$, где $|R|$ и $|S|$ – размер входных коллекций.

$0 \leq c_s \leq c_{all} \Rightarrow c_s \approx \frac{c_{all}}{2}$

$0 \leq c_p \leq c_s \Rightarrow c_p \approx \frac{c_s}{2}$

$0 \leq c_v \leq c_p \Rightarrow c_v \approx \frac{c_p}{2}$

$1 \leq l_{pr} \leq l_{ob} \Rightarrow l_{pr} \approx \frac{l_{ob} + 1}{2}$

Количество неизвестных параметров, выражаемых друг через друга, ухудшает точность оценки в несколько раз, что делает погрешность недопустимо высокой. Возможным решением данной проблемы является накопление статистических данных. Количество пар, остающихся после проверок, очевидным образом зависит от изначальных размеров коллекций, однако остающиеся части не имеют такой зависимости, что делает их сравнимым параметром для любых входных данных.

$c_s = x \cdot c_{all}, 0 \leq x \leq 1$

$c_p = y \cdot c_s, 0 \leq y \leq 1$

$c_v = z \cdot c_p, 0 \leq z \leq 1$

$l_{pr} = p \cdot l_{ob}, \frac{1}{l_{ob}} \leq p \leq 1$

Реализованный алгоритм подсчитывает количество пар остающихся после каждой проверки и среднюю длину префикса и хранит соответствующие значения x, y, z, p. Итоговые значения параметров вычисляются, как выборочное среднее накопленных данных.

Тогда при заданном времени работы алгоритма t_0, количество пар, по достижению которого алгоритму необходимо завершить свою работу, может быть выражено по следующей формуле:

$$c_{all} = \frac{\frac{t_0}{t_{sys}} - k(5 + 2l_{ob} + logk)}{3 + x \cdot (p^2 \cdot l_{ob}^2 + 1) + x \cdot y((2 - p)l_{ob} + 2) + x \cdot y \cdot z(2 + k)}$$

При этом среднее время одной вычислительной операции и средняя длина объекта являются статистическими данными, оцениваемыми для индивидуального оборудования и индивидуальной коллекции объектов.

6.2. Качество результата

При ограниченном времени работы, алгоритм возвращает не точный, а приближенный ответ. Для возможности оптимизации запроса, необходимо иметь конкретные оценки качества результата в зависимости от времени. Под качеством результата в нашем случае будем понимать точность. Обозначим через k' количество пар из возвращаемых k, принадлежащих истинным k лучшим результатам. Тогда $\frac{k'}{k}$ - точность возвращаемого результата.

Интуитивно понятно, что качество результата возрастает с увеличением количества обработанных пар. Если обозначить за $c_{all}^{'}$ количество пар, которых обрабатывает алгоритм за некоторое заданное время t_0, то точность ответа $\frac{k'}{k}$ зависит от отношения $\frac{c_{all}^{'}}{c_{all}}$. Логичным предположением было бы хранить коэффициент пропорциональности этих двух величин, как в случае с остальными статистическими параметрами, однако вследствие экспериментов такой подход показал себя не лучшим в конкретном случае.

Проблема возникает из-за того, что количество истинных пар k' в буфере возрастает с ростом количества пар c_{all} не равномерно. Алгоритм обрабатывает пары по мере убывания оценок объектов, что, в свою очередь, уменьшает вероятность получения

высокой итоговой оценки с течением времени. Таким образом, скорость нахождения истинных пар тоже падает. На рис. 2 представлен график, отражающий данную зависимость.

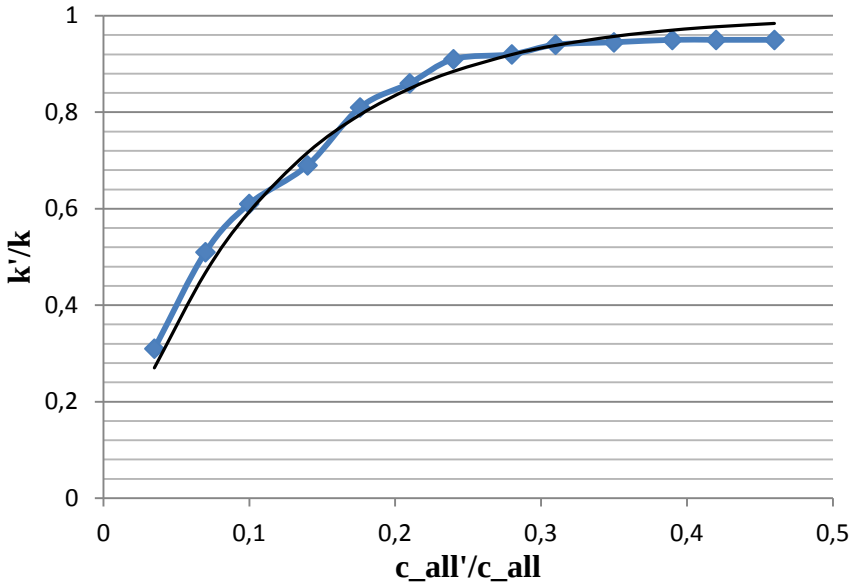

Рис. 2. Распределение точности результата $\frac{k'}{k}$ в зависимости от отношения количества обработанных пар приближенной и точной версии алгоритма $\frac{c_{all}^{'}}{c_{all}}$.

В результате, для оценки качества результата была выбрана экспоненциальная зависимость:

$\frac{k'}{k} = 1 - e^{a \cdot \frac{c_{all}^{'}}{c_{all}}}$, где a – хранимый статистический параметр.

При имеющемся значении параметра a, неизвестным заранее остается только параметр c_{all}, отражающий число пар, которых необходимо обработать для получения точного результата. Грубая оценка, полученная через математическое ожидание, уже была представлена выше, и, точно так же, его можно оценить с помощью хранения статистики, как часть исходного количества пар объектов.

$$c_{all} = w \cdot (|R| \cdot |S| - k), 0 \leq w \leq 1$$

Таким образом, зная значения статистических данных, мы можем заранее предсказать ожидаемую точность результата в зависимости от заданного времени. Данные оценки позволяют спланировать оптимальную стратегию поведения в каждом индивидуальном случае, управляя соотношением время/качество.

7. Эксперименты

Основной целью поставленных экспериментов являлось тестирование точности приведенной модели стоимости, а именно двух основных аспектов – предполагаемого качества результата и времени работы приближенной версии алгоритма.

В качестве тестовых наборов данных были сгенерированы две коллекции по 5000 и 10000 объектов. Объектами являлись текстовые строки, длиной от 10 до 55 слов. Подобие строк возникало благодаря тому, что все слова были выбраны в произвольном порядке из небольшой коллекции в 40 слов, где некоторые слова присутствовали в нескольких экземплярах. Оценки объектов так же генерировались произвольно в пределах промежутка [0.4, 1].

Алгоритм был реализован на C++. Предварительно объекты коллекций были разбиты на множества токенов и отсортированы в соответствии с возрастанием частоты документа. Разбивка производилась по таким разделительным символам, как пунктуация и пробелы. В качестве токенов выступали слова, при этом каждое повторное слово в одном объекте считалось новым токеном.

В качестве первого теста было проведено сравнение реального времени работы алгоритма и эталонного заданного времени. Разница получается вследствие приблизительного расчета количества пар, которых можно успеть обработать за это время, и является показателем точности представленной модели стоимости.

Результаты эксперимента продемонстрированы на рис. 3. Была проведена серия тестов при различных значениях k от 10 до 8500. Все параметры, описанные в разделе 6, были выражены двумя способами – грубо оценены через математическое ожидание и рассчитаны с учетом накопленной статистики. В реализацию алгоритма включено автоматическое сохранение статистики, модифицирующее и обновляющее параметры при каждом новом запуске, однако для чистоты эксперимента все тесты были проведены при одних и тех же значениях параметров.

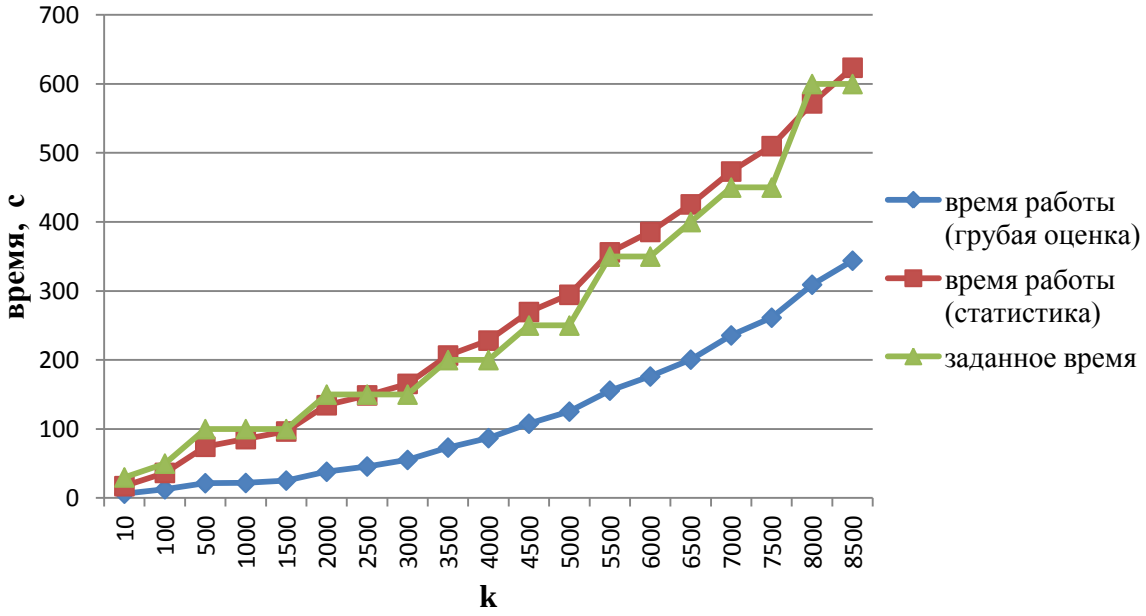

Рис. 3. Заданное и реальное время работы алгоритма

По приведенному графику можно отметить, что хотя при грубой оценке параметров реальное время алгоритма значительно отличается от требуемого, при имеющейся статистике результат близок к желаемому. Для оценки средних статистических параметров на рассматриваемом промежутке, статистика была собрана как для точной, так и для приближенной версии алгоритма, при k равных 10, 5000 и 10000.

Еще одним важным аспектом разработанной модели стоимости является предполагаемое качество результата. Качество оценивается по шкале [0, 1], как описано выше в разделе 6. Для оценки правдоподобности ожидаемого результата был проведен ряд тестов при значении k равном 5000 и различном заданном времени работы. Результаты эксперимента приведены на рис. 4. Как и в первом эксперименте, параметры здесь оценены как грубо, так и с учетом накопленной статистики.

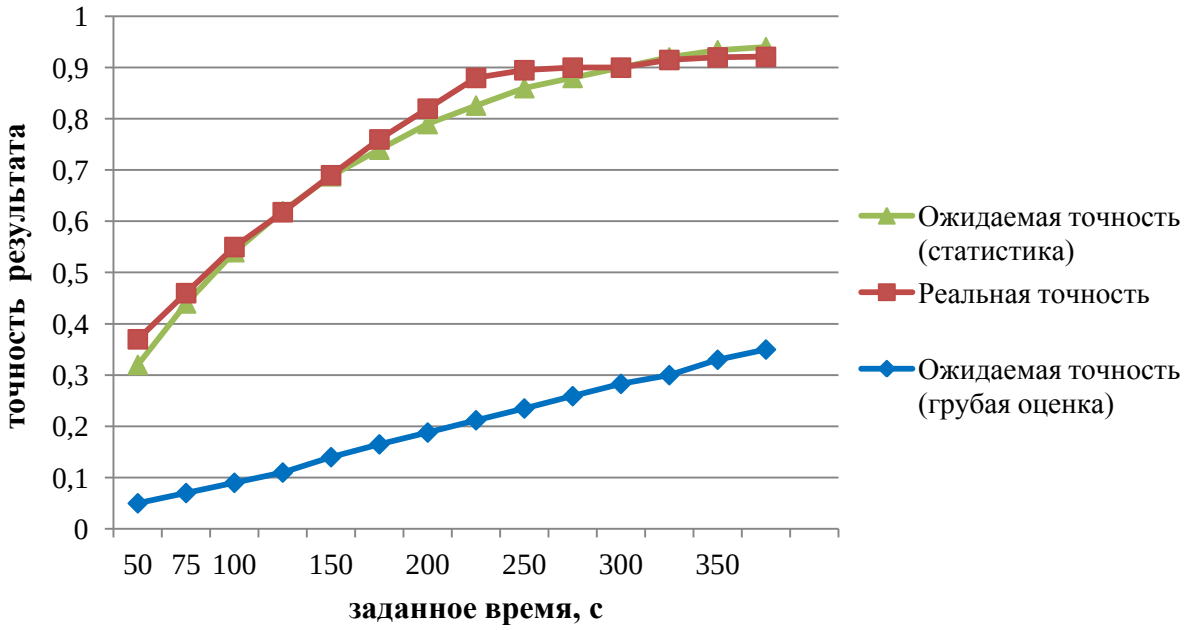

Рис. 4. Ожидаемое и реальное качество результата

По поставленным экспериментам видно, что грубой оценки недостаточно и для получения адекватных результатов необходимо иметь хоть какую-то статистику. Конечно, лишние запуски алгоритма с целью сбора статистических данных создают определенные неудобства.

Точность приведенной модели стоимости можно было бы улучшить, отказавшись от дополнительных проверок, таких как префиксное фильтрование и фильтрование по размеру, и сократив тем самым количество оцениваемых параметров. Чтобы проверить, насколько существенны данные проверки, было приведено сравнение реализованного алгоритма и наивного алгоритма соединения вложенными циклами, вычисляющего подобие каждой пары вплоть до условия остановки. Условие остановки при этом остается таким же, как и в описанном в разделе 5 алгоритме.

Результаты эксперимента показаны на рис. 5. Тесты были проведены на двух сгенерированных коллекциях размером 5000 и 1000 объектов. Как можно убедиться, при отсутствии этих двух проверок время работы алгоритма возрастает в разы. В ряде случаев даже если запускать алгоритм перед каждым запросом для сбора статистики, суммарное время работы все равно будет меньше, чем время работы наивного алгоритма.

Принимая же во внимание тот факт, что дополнительные запуски необходимы только перед первым запросом, когда статистика еще совершенно отсутствует, можно

смело сказать, что значительное сокращение затрачиваемого времени, привнесенное методами фильтрования, компенсирует возникающие неточности модели стоимости.

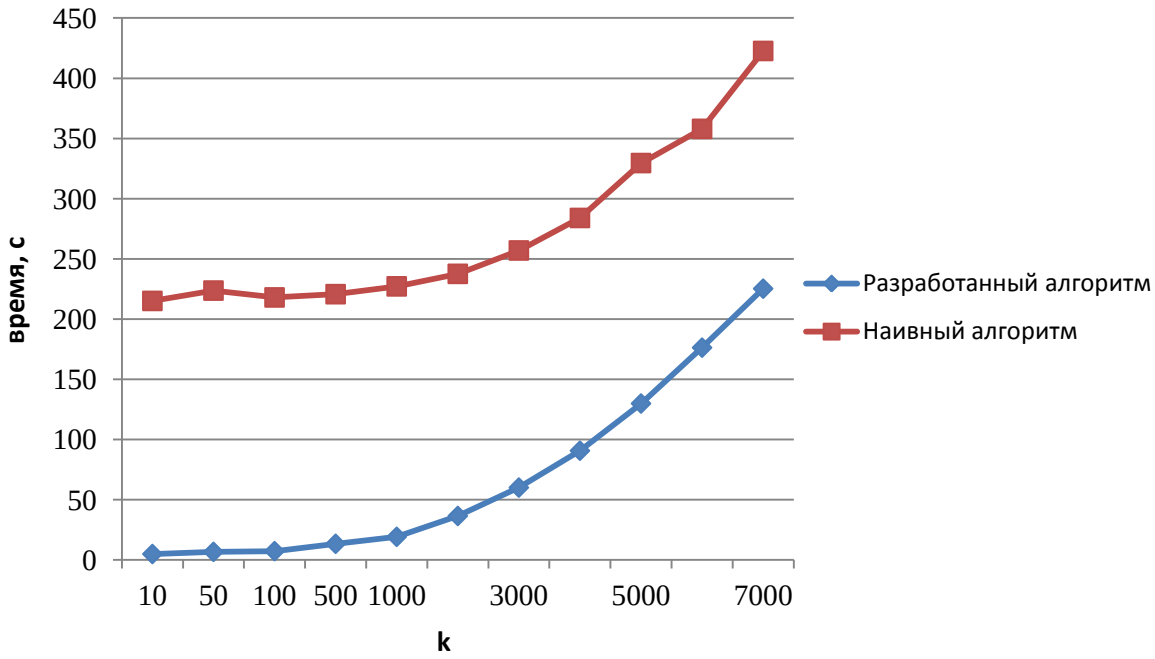

Рис. 5. Время работы реализованного и наивного алгоритма

8. Заключение

В результате работы был разработан и реализован алгоритм соединения по подобию на множествах объектов с оценками. В алгоритме использованы известные техники префиксного фильтрования и фильтрования по размеру [14], позволяющие эффективно сократить набор кандидатов без вычисления значения подобия пар и в разы уменьшить время выполнения операции.

Разработанный алгоритм носит приближенный характер и принимает отведенное время работы, как параметр. Качество результата зависит от заданного времени, при отсутствии же этого параметра алгоритм возвращает точные результаты.

В работе так же была построена расширенная модель стоимости, позволяющая заранее оценить точность предполагаемого результата. Сведения о соотношении заданного времени и обещанного качества предоставляют возможность управления им для оптимизации данной операции. После этого операция может быть использована, как часть более сложных запросов.

Были проведены эксперименты, в ходе которых была протестирована точность приведенной модели с грубой и статистической оценкой имеющихся в модели параметров. Разработанная модель показала себя работоспособной и достаточно точной при сохранении статистики. Так же результаты экспериментов демонстрируют оправданность данного неудобства.

9. Список литературы

1. *A. Arasu, V. Ganti, and R. Kaushik.* Efficient exact set-similarity joins. // VLDB, 2006.
2. *Benjamin Arai, Gautam Das, and Dimitrios Gunopulos*. Anytime Measures for Top-k Algorithms. // VLDB, 2007.
3. *Boris Novikov, Natalia Vassilieva, and Anna Yarygina*. Querying big data. // CompSysTech, 2012.
4. *Chuan Xiao, Wei Wang, and Xuemin Lin*. Ed-Join: An Efficient Algorithm for Similarity Joins With Edit Distance Constraints. // PVLDB, 1(1):933–944, 2008.
5. *Chuan Xiao, Wei Wang, and Xuemin Lin*. Efficient Similarity Joins for Near Duplicate Detection. // WWW, 2008.
6. *Chuan Xiao, Wei Wang, Xuemin Lin, and Haichuan Shang*. Top-k Set Similarity Joins. // ICDE, 2009.
7. *Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos*. Closest Pair Queries in Spatial Databases. // SIGMOD Conference, 2000, pp. 189–200.
8. *Gionis, P. Indyk, and R. Motwani*. Similarity search in high dimensions via hashing. // VLDB, 1999.
9. *Jiannan Wang, Jianhua Feng, and Guoliang Li*. Trie-Join: Efficient Triebased String Similarity Joins with EditDistance Constraints. // PVLDB, 3(1):1219–1230, 2010.
10. *Jiannan Wang, Guoliang Li, and Jianhua Feng*. Can we beat the prefix filtering?: an adaptive framework for similarity join and search. // SIGMOD, 2012.
11. *M. Charikar*. Similarity estimation techniques from rounding algorithms. // STOC, 2002.
12. *Martin Theobald, Gerhard Weikum, and Ralf Schenkel*. Top-k Query Evaluation with Probabilistic Guarantees. // Proceedings of VLDB, 2004.
13. *R. Fagin, A. Lotem, and M. Naor*. Optimal aggregation algorithms for middleware. // J. Comput. Syst. Sci., vol. 66, no. 4, pp. 614–656, 2003.
14. *R. J. Bayardo, Y. Ma, and R. Srikant*. Scaling up all pairs similarity search. // WWW, 2007.
15. *S. Chaudhuri, V. Ganti, and R. Kaushik.* A primitive operator for similarity joins in data cleaning. // ICDE, 2006.
16. *Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig*. Syntactic clustering of the web. // Computer Networks, vol. 29, no. 8-13, pp. 1157–1166, 1997.