МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Технология программирования
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Применение методов нечеткого сравнения строк в прикладных задачах»

Студент Д.Л. Ниёзов
(И.О. Фамилия) (личная подпись)

Руководитель М.А. Тренина
(ученая степень, звание, И.О. Фамилия)

Консультант О. А. Головач
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Аннотация

Тема выпускной квалификационной работы – «Применение методов нечеткого сравнения строк в прикладных задачах».

Исследование применения методов нечеткого сравнения строк при решении различных прикладных задач представляет актуальность и научно-практический интерес.

Объектом исследования бакалаврской работы являются методы нечеткого сравнения строк.

Предметом исследования выпускной квалификационной работы является применение методов нечеткого сравнения строк для решения прикладных задач.

Целью выпускной квалификационной работы является изучение методов нечеткого сравнения строк и исследование их применения при разработке программного обеспечения для решения различных прикладных задач.

В выпускной квалификационной работе рассмотрены методы Дамерау-Левенштейна и Джаро-Винклера, проведён краткий анализ и сравнение.

Проанализированы свойства алгоритмов Вагнера-Фишера, Дамерау-Левенштейна и Джаро-Винклера, а также рассмотрены их достоинства и недостатки.

Разработана программа, в которой реализован алгоритм Джаро-Винклера для поиска клиентов в списке Росфинмониторинга и продемонстрирована на реальных данных.

Результаты бакалаврской работы могут быть рекомендованы для разработчиков программного обеспечения, использующего методы нечеткого сравнения строк.

Структура бакалаврской работы: 43 страницы, 14 рисунков, 2 таблицы, 25 источников.

Abstract

The topic of the given graduation work is Utilizing methods of fuzzy string matching in applied tasks.

The study of the application of fuzzy string matching methods in solving various applied tasks is of relevance and scientific and practical interest.

The object of study of the graduation work are fuzzy string matching methods.

The subject of study of the graduation work is utilizing methods of fuzzy string matching in applied tasks.

The aim of the graduation work is to study of fuzzy string matching methods and to research of their application in the development of software for solving various applied tasks.

In the graduation work methods of Damerau-Levenshtein and Jaro-Winkler are considered, a brief analysis and comparison is carried out.

The properties of the Wagner-Fischer, Damerau-Levenshtein and Jaro-Winkler algorithms are analyzed, and their advantages and disadvantages are considered.

A program has been developed that implements the Jaro-Winkler algorithm for searching customers in the Rosfinmonitoring list and is demonstrated on real data.

The results of bachelor's work can be recommended for developers of software using the fuzzy string matching methods.

The graduation work consists of an explanatory note on 43 pages including 14 figures, 2 tables, the list of 25 references.

Оглавление

Введение

В компьютерных науках нечеткое сравнение строк – это поиск строк в тексте, которые приблизительно соответствуют заданному шаблону.

Другими словами, нечеткое сравнение строк – это тип текстового поиска, который находит совпадения, даже если пользователи неправильно вводят слова или вводят только частичные слова для поиска.

Это задача, известная также, как приблизительное совпадение строк, широко применяется в поисковых системах, базах данных, системах проверки орфографии и на плагиат, а также в других прикладных областях [20].

Так, при разработке многоступенчатых фильтров в системах управления базами данных на одной из стадий необходимо выполнить приблизительное сравнение строк [6]. Как показывает практика, эффективность данной стадии зависит от грамотно подобранных критериев идентичности строк и алгоритмов их сравнения.

Для решения таких задач используются методы нечеткого сравнения строк.

Исследование применения методов нечеткого сравнения строк при решении различных прикладных задач представляет **актуальность** и научно-практический интерес.

Объектом исследования бакалаврской работы являются методы нечеткого сравнения строк.

Предметом исследования бакалаврской работы является применение методов нечеткого сравнения строк для решения прикладных задач.

Цель выпускной квалификационной работы – анализ алгоритмов нечеткого сравнения строк и исследование возможности их применения при разработке программного обеспечения для решения различных прикладных задач.

Для достижения данной цели необходимо выполнить следующие задачи:

– провести анализ методов нечеткого сравнения строк;

– провести анализ известных алгоритмов нечеткого сравнения строк;

– исследовать применение алгоритмов нечеткого сравнения строк в различных прикладных задачах;

– разработать программу реализации алгоритма нечеткого сравнения строк и оценить ее эффективность.

Методы исследования – методы и алгоритмы нечеткого сравнения строк, объектно-ориентированный подход к проектированию программного обеспечения, теория баз данных.

Практическая значимость бакалаврской работы заключается в разработке программы, реализующей алгоритм нечеткого сравнения строк.

Данная работа состоит из введения, трех глав, заключения, списка используемой литературы и приложений.

Первая глава посвящена описанию методов нечеткого сравнения строк. Рассмотрены методы Дамерау-Левенштена и Джаро-Винклера.

Во второй главе дан обзор известных алгоритмов нечеткого сравнения строк. Описаны свойства алгоритмов Вагнера-Фишера, Дамерау-Левенштена и Джаро-Винклера. Представлен сравнительный анализ указанных алгоритмов.

В третьей главе рассмотрены примеры применения алгоритмов нечеткого сравнения строк для решения различных прикладных задач. Разработана программа, использующая алгоритм Джаро-Винклера для нечеткого сравнения строк в процессе поиска клиентов в Перечне Росфинмонитринга.

В заключении описываются результаты выполнения выпускной квалификационной работы.

Структура бакалаврской работы: 43 страницы, 14 рисунков, 2 таблицы, 25 источников.

Глава 1 Методы нечеткого сравнения строк

Основная идея методов нечеткого сравнения строк заключается в том, чтобы предоставить пользователю возможность определить, что два текста (или наборы данных в целом) полностью или хотя бы частично похожи между собой.

Для исследования методов нечеткого сравнения строк необходимо предварительно описать понятия меры и метрики строки [9].

Меры на основе строк работают с последовательностями строк и композицией символов.

В математике и информатике строковая метрика (также известная как метрика сходства строк или функция расстояния строк) – это метрика, которая измеряет расстояние («обратное сходство») между двумя текстовыми строками для приблизительного сопоставления или сравнения строк и при поиске нечеткой строки.

Для определения сходства строк используются следующие подходы [22]:

- основанный на символах (Character-based);
- основанный на терминах (Term-based);
- основанный на корпусной лингвистике (Corpus-based);
- основанный на знаниях (Knowledge-based).

Наиболее популярными и широко используемыми в различных прикладных задачах являются методы нечеткого сравнения строк, основанные на символьном подходе.

Рассмотрим данные методы.

1.1 Метод Дамерау-Левенштейна

Метод основан на понятии расстояния Левенштейна.

В теории информации, лингвистике и информатике расстояние Левенштейна является строковой метрикой для измерения разницы между двумя последовательностями.

Иначе расстояние Левенштейна называется расстоянием редактирования и рассматривается как способ количественной оценки того, насколько разнородны две строки (например, слова), путем подсчета минимального количества операций, необходимых для преобразования одной строки в другую.

Таким образом, неформально расстояние Левенштейна между двумя словами – это минимальное количество односимвольных правок (вставок, удалений или подстановок), необходимых для замены одного слова на другое [3].

Рассмотрим определение расстояния Левенштейна.

Расстояние Левенштейна между двумя строками *a,b* (длиной |a| и |b|, соответственно) $\mathrm{lev}_{a,b}(|a|,|b|)$ определяется из выражения:

$$\mathrm{lev}_{a,b}(i,j)=\begin{cases}\max(i,j) & \text{если } \min(i,j)=0\\ \min\begin{cases}\mathrm{lev}_{a,b}(i-1,j)+1\\ \mathrm{lev}_{a,b}(i,j-1)+1\\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{a_i\neq b_j}\end{cases} & \text{в противном случае,}\end{cases} \quad (1)$$

где:

$\mathrm{lev}_{a,b}(i,j)$ – расстояние между первыми i символами строки a и первыми j символами строки b (i и j – индексы индикаторной функции).

$1_{a_i\neq b_j}$ – индикаторная или характеристическая функция, равная 0, если $a_i=b_j$ и равна 1 в противном случае;

Следует отметить, что первый элемент минимума соответствует удалению (от *a* к *b*), второй – вставке, а третий – совпадению или

несовпадению, в зависимости от того, совпадают ли соответствующие символы.

Расстояние Дамерау-Левенштейна отличается от расстояния Левенштейна включением транспонирования в число допустимых операций в дополнение к трем классическим односимвольным операциям редактирования (вставки, удаления и замены) [16].

Чтобы выразить расстояние Дамерау-Левенштейна между двумя строками *a* и *b* используется функция $d_{a,b}(i,j)$, значение которой является расстоянием между *i*-символьным префиксом (исходной подстрокой) строки a и *j*-символьным префиксом строки b:

$$d_{a,b}(i,j) = \min \begin{cases} 0 & \text{если } i = j = 0 \\ d_{a,b}(i-1, j) & \text{если } i > 0 \\ d_{a,b}(i, j-1) & \text{если } j > 0 \\ d_{a,b}(i-1, j-1) + 1_{a_i \neq b_j} & \text{если } i, j > 0 \\ d_{a,b}(i-2, j-2) + 1 & \text{если } i, j > 1 \text{ и } a[i] = b[j-1] \text{ и } a[i-1] = b[j] \end{cases} \quad (2)$$

где:

$1_{a_i \neq b_j}$ – индикаторная функция, равная 0, если $a_i = b_j$ и равна 1 в противном случае.

Каждый рекурсивный вызов соответствует одному из случаев, охватываемых расстоянием Дамерау-Левенштейна:

$d_{a,b}(i-1,j)$ соответствует удалению (от адо b);

$d_{a,b}(i,j-1)$ соответствует вставке (от адо b);

$d_{a,b}(i-1,j-1)$ соответствует совпадению или несовпадению, в зависимости от того, совпадают ли соответствующие символы;

$d_{a,b}(i-2,j-2)+1$ соответствует транспозиции (перестановке) между двумя последовательными символами.

Расстояние Дамерау-Левенштейна между *a* и *b* затем определяется значением функции для полных строк: $d_{a,b}(|a|,|b|)$, где *i*=|a|, обозначает длину строки a, *j*=|b|–длину строки b.

1.2 Метод Джаро-Винклера

Данный метод основан на понятии расстояния Джаро [19].

Расстояние Джаро – это мера сходства между двумя строками.

Чем выше расстояние Джаро для двух строк, тем больше сходства между ними.

Оценка нормализуется таким образом, что 0 означает отсутствие сходства, а 1 - точное совпадение.

Определение: расстояние Джаро для двух строк s_1 и s_2 определяется из выражения:

$$d_j = \min \begin{cases} 0 & \text{если } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{в противном случае,} \end{cases} \qquad (3)$$

где:

m – количество совпадающих символов;

t – количество транспозиций.

Два символа s_1 и s_2 соответственно считаются совпадающими, только если они одинаковы и расположены не дальше, чем:

$$\frac{\max(|s_1|,|s_2|)}{2} - 1 \qquad (4)$$

Каждый символ s_1 сравнивается со всеми соответствующими символами в s_2.

Количество совпадающих (но различающихся порядков последовательности) символов, разделенное на 2, определяет количество транспозиций.

Рассмотрим метод Джаро-Винклера [25].

В методе Джаро-Винклера используется шкала префикса *p*, которая дает более благоприятные оценки для строк, соответствующие заданной длине префикса *l* в начале строки.

Для двух строк s_1 и s_2 сходство Джаро-Винклера определяется из выражения:

$$\mathrm{sim}_w = \mathrm{sim}_j + lp(1-\mathrm{sim}_j), \tag{5}$$

где:

sim_j – расстояние Джаро для строк s_1 и s_2;

l – длина общего префикса в начале строки, не более четырех символов;

p – постоянный коэффициент масштабирования для увеличения оценки за наличие общих префиксов.

Величина *p* не должна превышать 0.25, в противном случае сходство может превысить 1.

Стандартное значение *p*=0.1.

Расстояние Джаро-Винклера определяется из выражения:

$$d_w = 1-\mathrm{sim}_w \tag{6}$$

Хотя это полученное расстояние часто называют метрикой расстояния, оно не является метрикой в математическом смысле этого термина, поскольку оно не подчиняется неравенству треугольника.

Следует также отметить, что расстояние Джаро-Винклера не удовлетворяет аксиоме тождества $d(x,y) = 0 \leftrightarrow x = y$.

Основные строковые метрики не учитывают семантику строк. Они основаны исключительно на определении того, насколько строки похожи между собой с точки зрения составляющих их символов.

В связи с этим, если не прибегать к нормализации, то строковые метрики будут малоэффективны.

Однако, если всё же нормализовать текстовые данные, а затем представить эти данные в виде атомарных атрибутов, семантические компоненты строк отделяются друг от друга. В результате простое определение расстояния между символами строк может стать источником ценной информации [7].

Строковые метрики не являются комплексным решением, но они играют важную роль в поиске нечетких соответствий – именно они позволяют нам определять, насколько схожи компоненты сущностей, а также автоматизировать процесс их сравнения и сократить степень условности такого сравнения.

Выводы к первой главе

1. Основная идея методов нечеткого сравнения строк заключается в определении степени сходства между двумя текстами или наборами данных. Для этого используется метрика строки или функция расстояния строк. Наиболее популярными считаются методы нечеткого сравнения строк, основанные на символьном подходе, так как считаются точными при сравнении строк.

2. В методе Дамерау-Левенштейна используется расстояние Левенштейна, которое является строковой метрикой для измерения разницы между двумя строковыми последовательностями.

3. Расстояние Дамерау-Левенштейна отличается от расстояния Левенштейна включением транспонирования в число допустимых операций

в дополнение к трем классическим односимвольным операциям редактирования (вставки, удаления и замены).

4. Метод Джаро-Винклера основан на понятии расстояния Джаро, которое, в свою очередь, используется как мера сходства между двумя строками. Чем выше расстояние Джаро для двух строк, тем больше сходства между ними.

5. В методе Джаро-Винклера используется шкала префикса, которая дает более благоприятные оценки для строк, соответствующие заданной длине префикса в начале строки.

6. Несмотря на то, что все вышеперечисленные методы хоть и имеют некоторые сходства, всё же используются для достижения различных результатов. Метрику Джаро-Винклера рекомендуется использовать для строк небольшой длины. Для длинных же строк лучше использовать метрику Дамерау-Левенштейна.

Глава 2 Анализ алгоритмов нечеткого сравнения строк

Алгоритмы нечеткого сравнения строк относятся к категории алгоритмов сравнения строк.

Сравнение строк – это способ, используемый для поиска результатов одного или нескольких заданных текстовых шаблонов.

Сравнение строк является важной задачей в области компьютерных наук, поскольку текст является основной формой обмена информацией между людьми, например, в литературе, научных работах и на веб-страницах.

Алгоритм сравнения строк – это алгоритм поиска всех вхождений строк путем поиска сходств.

Принцип соответствия строк состоит в том, чтобы найти все вхождения коротких строк $P[0,\ldots,n\text{-}1]$, называемых шаблонами, в более длинную строку текста $T[0,\ldots,m\text{-}1]$, где n и m – длины сравниваемых строк.

Рассмотрим формальное определение алгоритма нечеткого поиска [4].

Пусть Σ – конечное множество (алфавит) размера $|\Sigma|=\sigma$.

Пусть $T \in \Sigma^*$ – текст длиной $n = |T|$.

Пусть $P \in \Sigma^*$ – шаблон длиной $m = |P|$.

Пусть $k \in \mathbf{R}$ – максимально разрешенное количество ошибок.

Пусть $d : \Sigma^* \times \Sigma^* \rightarrow \mathbf{R}$ – функция расстояния.

Задача: даны T, P, k и d(.), которые возвращают множества всех позиций текста j таких, что существует i, для которого $d(P, T_{i..j}) \leq k$[2].

2.1 Алгоритм Вагнера-Фишера

Алгоритм Вагнера-Фишера – это алгоритм динамического программирования, позволяющий вычислить расстояние редактирования между двумя символьными строками по методу Левенштейна [23].

Алгоритм Вагнера-Фишера позволяет вычислить расстояние редактирования на основе наблюдения, что, если зарезервировать матрицу для хранения расстояний редактирования между всеми префиксами первой строки и всеми префиксами второй, то можно вычислить значения в матрице, путем заполнения матрицы, и, таким образом, найти расстояние между двумя полными строками в качестве последнего вычисленного значения.

Алгоритм Вагнера-Фишера состоит из следующих шагов:

Шаг 1. Задать строки s и t;

Шаг 2. Определить длины строк: N=len(s), M=len(t);

Шаг 3. Если M=0, возврат N и выход,

Если N=0, возврат M и выход;

Шаг 4. Создать матрицу Mat[M*N]

Шаг 5. Инициализировать 1-ю строку и -1 столбец матрицы: 0,…,Nи 0,…,M;

Шаг 6. Исследовать каждый символ строки s для i от 1 до N,

Исследовать каждый символ строки t для j от 1 до M

Шаг 7. Если s[i]==t[j], тогда cost=1,

В противном случае cost=0

Шаг 8. Установить ячейку d[i,j] матрицы равной минимуму:

а) ячейка непосредственно сверху плюс **1**:d[i-1,j]+1

б) ячейка непосредственно слева плюс **1**:d[i,j-1] +1

в) ячейка по диагонали сверху плюс cost:d[i-1,j-1] +cost;

Шаг 9. После итерации шагов 3-6 определяем расстояние Левенштейна в ячейке d[N,M].

Простая реализация алгоритма Вагнера-Фишера в виде псевдокода для функции LevenshteinDistance, которая принимает две строки s длиной m и t длины n и возвращает расстояние Левенштейна между ними, имеет следующий вид:

function LevenshteinDistance(char s[1..m], char t[1..n]):

// Для всех i and j, d[i,j] будет соблюдаться расстояние Левенштейна

//между первыми i символами строки s и первыми j символами строки t

// следует иметь в виду, что d имеет (m+1)*(n+1)значений

declare int d[0..m, 0..n]

set each element in d to zero

// исходные префиксы можно преобразовать в пустую строку,

// удалив все символы

forifrom 1 tom:

d[i, 0] := i

// целевые префиксы могут быть достигнуты из пустого исходного

//префикса путем вставки каждого символа

forjfrom 1 ton:

d[0, j] := j

```
  for j from 1 to n:
    for i from 1 to m:
      if s[i] = t[j]:
        substitutionCost := 0
      else:
        substitutionCost := 1
      d[i, j] := minimum(d[i-1, j] + 1,                  // удаление
                         d[i, j-1] + 1,                  // вставка
                         d[i-1, j-1] + substitutionCost)  // замена
```

return d[m,n]

Следует обратить внимание на то, что начальный индекс входных строк равен 1, а матрица *d* индексируется, начиная с нуля, причем [i..k] является закрытым диапазоном.

Инвариант, поддерживаемый на протяжении всего алгоритма, заключается в том, что мы можем преобразовать начальный сегмент s[1..i] в t [1..j], используя минимум операций d [i, j].

Рассмотрим пример вычисления расстояния редактирования с помощью алгоритма Вагнера-Фишера для двух строк: A = "acat" иB = "anact".

Расстояние Левенштейна между ними равно 3: для перехода от A к B требуется одна вставка ('n') и две замены ('a' на 'c' и 'c' на 'a').

Результирующая таблица Вагнера-Фишера представлена на рисунке 1.

		a	n		a	c	t
	0	1	2	3	4	5	6
a	1	0	1	2	3	4	5
	2	1	1	1	2	3	4
c	3	2	2	2	2	2	3
a	4	3	3	3	2	3	3
t	5	4	4	4	3	3	3

Рисунок 1 - Результирующая таблица алгоритма Вагнера-Фишера

Нижний правый элемент массива содержит ответ.

Алгоритм Вагнера-Фишера очень требователен к памяти.

Неоптимизированный вариант этого алгоритма имеет временную сложность $O(m \cdot n)$ и потребляет $O(m \cdot n)$ памяти, где m и n – длины сравниваемых строк.

Таким образом, данный алгоритм крайне неэффективен при сравнении длинных строк.

2.2 Алгоритм Дамерау-Левенштейна

Алгоритм Дамерау-Левенштейна позволяет вычислить расстояние редактирования на основе одноименного метода.

Рассмотрим версию алгоритма Дамерау-Левенштейна, позволяющего вычислить так называемое оптимальное расстояние выравнивания строки или ограниченное расстояние редактирования.

Реализация данного алгоритма в виде псевдокода имеет вид:

```
algorithm OSA-distance is
input: strings a[1..length(a)], b[1..length(b)]
output: distance, integer
let d[0..length(a), 0..length(b)] be a 2-d array of integers, dimensions length(a)+1, length(b)+1
// dимеет 0-йначальныйиндекс, аиb индексируются с 1.
for i := 0 to length(a) inclusive do
d[i, 0] := i
   for j := 0 to length(b) inclusive do
d[0, j] := j
      for i := 1 to length(a) inclusive do
for j := 1 to length(b) inclusive do
         if a[i] = b[j] then
            cost := 0
         else
            cost := 1
            d[i, j] := minimum(d[i-1, j] + 1, // удаление
d[i, j-1] + 1, // вставка
                              d[i-1, j-1] + cost)
                                    // замена
            if i > 1 and j > 1
               and a[i] = b[j-1]
               and a[i-1] = b[j] then
               d[i, j] := minimum(d[i, j],
                                 d[i-2, j-2] + 1)
                                       // транспозиция
```

return d[length(a), length(b)]

Отличие от стандартного алгоритма расстояния Левенштейна заключается в добавлении рекурсии, выделенной курсивом.

Результирующая таблица для рассмотренного выше случая будет иметь вид, представленный на рисунке 2 [17].

		a		a	b	c	t
	0	1	2	3	4	5	6
a	1	0	1	2	3	4	5
	2	1	0	1	2	3	4
c	3	2	1	1	2	2	3
a	4	3	2	1	2	2	3
t	5	4	3	2	2	3	2

Рисунок 1 - Результирующая таблица алгоритма Дамерау-Левенштейна

Таким образом, алгоритм позволяет получить оптимальное значение расстояния редактирования, равное 2.

Временная сложность алгоритма Дамерау-Левенштейна– $O(m \cdot n \cdot \max(m,n))$.

Соответственно затраты памяти составляют $O(m \cdot n)$.

Вместе с тем, модификации данного алгоритма позволяют увеличить скорость работы алгоритма.

2.3 Алгоритм Джаро-Винклера

Алгоритм расстояния Джаро-Винклера – это алгоритм, использующий подход с метрикой строк, который выполняет сравнение строк с помощью соответствующих математических функций.

Следует отметить, что алгоритм Джаро-Винклера – это алгоритм измерения сходства между двумя строками, и большая часть этого алгоритма используется для обнаружения дублирования в строках.

Чем выше значение расстояния Джаро-Винклера для двух строк, тем выше сходство обеих строк.

Нормальным значением является 0, которое указывает на отсутствие сходства, и 1, которое указывает на существование точного сходства.

Алгоритм Джаро-Винклера для двух строк состоит из следующих основных фаз:

Фаза 1. Расчет длин строк.

Фаза 2. Поиск совпадающих символов в строках.

Фаза 3. Определение количества транспозиций.

Блок-схема алгоритма Джаро-Винклера представлена на рисунке 3 [14].

Рисунок 3 – Блок-схема алгоритма Джаро-Винклера

Рассмотрим пример выполнения алгоритма Джаро-Винклера при сравнении строк MARTHA и MARHTA [15].

В данном случае расстояние составляет max(6,6) / 2 - 1 = 6/2 - 1 = 2, что позволяет сопоставить не более двух символов.

Таблица выполнения алгоритма будет иметь вид (таблица 1):

Таблица 1 – Таблица выполнения алгоритма Джаро-Винклера

S1	M	A	R	**T**	**H**	A
Совпадения	0	1	2	4	3	5
S2	M	A	R	**H**	**T**	A

Следует учесть, что Т на позиции 3 в первой строке совпадает с Т на позиции 4 во второй строке, тогда как Н на позиции 4 в первой строке совпадает с Н на позиции 3 во второй строке.

Строки, которые не выровнены напрямую, отображаются жирным шрифтом. Это пример транспонирования.

Количество половин транспозиций определяется путем сравнения подпоследовательностей первой и второй совпадающих строк, а именно MARTHA и MARHTA.

Есть две позиции с несовпадающими символам –это 3 и 4. Это приводит к двум половинным транспозициям или одиночной транспонирования для расстояния Джаро:

1) расстояние_Джаро (MARTHA,MARHTA) = 1/3 * (6/6 + 6/6 + (6 - 1)/6) = 0.944.

2) Три начальных символа MAR в обеих строках совпадают. Тогда получим:

расстояние_Джаро (MARTHA, MARHTA) = 1 + 0,1 * 3 * (1,0 - 0,944) = 0.961.

Временная сложность стадии предварительной обработки алгоритма Джаро-Винклера – $O(m)$.

Временная сложность стадии поиска – $O(n)^2$.

Требуемая временная сложность – $O(m+ n)^2$.

Как показывает практика, алгоритм Джаро-Винклера обладает высокой точностью при сравнении коротких строк.

2.4 Сравнительный анализ алгоритмов нечеткого сравнения строк

Для сравнения представленных алгоритмов создадим таблицу их достоинств и недостатков (таблица 2).

Таблица 2 – Сравнительный анализ алгоритмов нечеткого сравнения строк

Алгоритм	Достоинства	Недостатки
Вагнер-Фишер	относительная простота реализации; возможность сравнения 2х и более строк; универсальность для различных алфавитов	низкая эффективность при сравнении длинных строк; большие расстояния при транспозиции слов или их частей
Дамерау-Левенштейн	соответствуют алгоритму Вагнера-Фишера; возможность получения оптимального расстояния выравнивания строки	соответствуют алгоритму Вагнера-Фишера
Джаро-Винклер	показывает хорошие результаты на естественных текстах	вычислительно сложен для длинных текстов

Таким образом, все представленные и проанализированные выше алгоритмы нечеткого сравнения строк обладают определенными достоинствами и недостатками.

Выбор того или иного алгоритма зависит в той или иной мере от конкретной прикладной задачи, а также требований по ее программной реализации.

Выводы ко второй главе

1. Алгоритмы нечеткого сравнения строк представляют собой алгоритмы, с помощью которых осуществляется поиск всех вхождений строк путем поиска их сходств.

2. Алгоритм динамического программирования Вагнера-Фишера позволяет вычислить расстояние редактирования между двумя символьными строками по методу Левенштейна.

3. Алгоритм Джаро-Винклера – это алгоритм измерения сходства между двумя строками, и большая часть этого алгоритма используется для обнаружения дублирования в строках. Чем выше значение расстояния Джаро-Винклера для двух строк, тем выше сходство обеих строк.

4. Все алгоритмы нечеткого сравнения строк обладают определенными достоинствами и недостатками. Выбор того или иного алгоритма зависит в той или иной мере от конкретной прикладной задачи и требования по программной реализации.

Глава 3 Применение алгоритмов нечеткого сравнения строк в прикладных задачах

Как показывает практика, алгоритмы нечеткого сравнения строк довольно широко применяются при решении различных прикладных задач.

Рассмотрим некоторые области применения данных алгоритмов.

3.1 Программа сверки справочников между базами 1С v8.1

Программа предназначена для сравнения наименований элементов справочников двух баз данных 1С8 [11].

Проблема актуальна при решении задач консолидации информационных баз данных конфигураций 1С (например, бухгалтерского и производственного учета), когда справочники в обеих конфигурациях ведутся раздельно.

Элементы сопоставляются по коду. При этом возможны отличия в наименованиях справочных записей.

Для приведения справочников в соответствие используется обработка, реализующая алгоритм Джаро-Винклера на языке 1С8. По результатам сверки формируется отчет, представленный на рисунке 4.

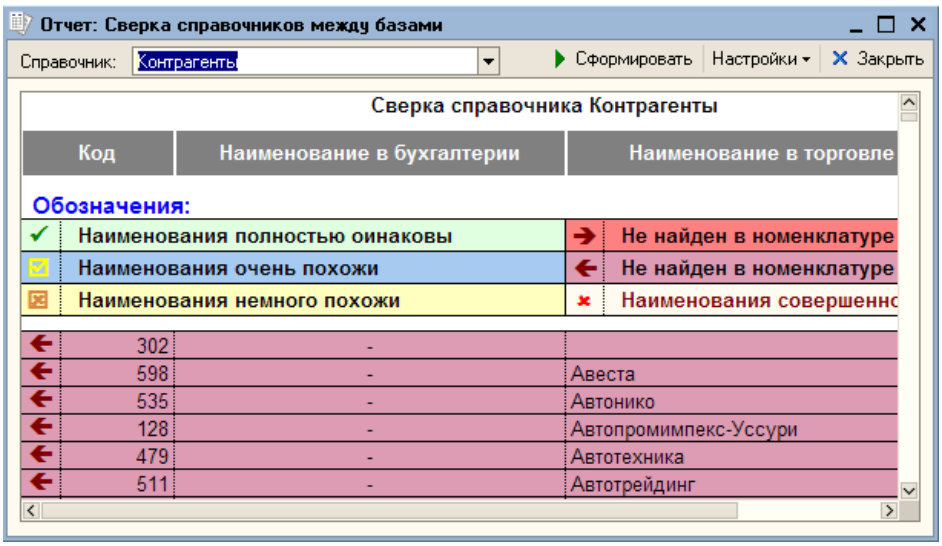

Рисунок 4 – Окно сверочного отчета

Программный код 1С8 универсальной функции для получения сходства Джаро-Винклера, представлен на рисунке 5 [12].

```
Функция ПолучитьСходствоДжароВинклера(Строка1,Строка2,Процент =
Ложь,ПерваяКороткая = Неопределено) Экспорт

        s1 = ВРег(СокрЛП(Строка1));
        s2 = ВРег(СокрЛП(Строка2));
        Если s1 = s2 Тогда
                Возврат ?(Процент,100,1);
        КонецЕсли;
        Если НЕ ПерваяКороткая = Неопределено
                И СтрДлина(s1) <> СтрДлина(s2) Тогда
                Буфер = Неопределено;
                Если ПерваяКороткая Тогда
                        Если СтрДлина(s1) > СтрДлина(s2) Тогда
                                Буфер = s1;
                                s1    = s2;
                                s2    = Буфер;
                        КонецЕсли;
                Иначе
                        Если СтрДлина(s2) > СтрДлина(s1) Тогда
                                Буфер = s1;
                                s1    = s2;
                                s2    = Буфер;
                        КонецЕсли;
                КонецЕсли;
                Буфер = Неопределено;
        КонецЕсли;

        // инициализация переменных
        a  = СтрДлина(s1);
        b  = СтрДлина(s2);
        d  = Цел(Макс(a,b)/2)-1;
        m  = 0;         tr = 0;
        t  = 0;
        L  = 0;
        p  = 0.1;
        bt = 0.7;
        dj = 0;
        dw = 0;
        bf = Новый Соответствие;
        Для Инд1 = 1 По a Цикл
        Символ1 = Сред(s1,Инд1,1);
                Для Инд2 = 1 По b Цикл
                                Если Инд2 < Инд1 - d Тогда
                                Продолжить;
                        КонецЕсли;
                        Если Инд2 > Инд1 + d Тогда
                                Прервать;
                        КонецЕсли;
```

```
                    Символ2 = Сред(s2,Инд2,1);
                    Если Символ1 = Символ2 Тогда
                    Если bf.Получить(Инд2) = Символ1 Тогда
                        Продолжить;
                    Иначе
                        bf.Вставить(Инд2,Символ1);
                        m = m + 1;
                        Если Инд1 <> Инд2 Тогда
                            tr = tr + 1;
КонецЕсли;
                    Если Инд1 = Инд2 Тогда
                            Если Инд1 = 1 Тогда
                                L = 1;
Иначе
                                Если L <= 3 И Инд1 = L + 1 Тогда
                                    L = L + 1; //
                                КонецЕсли;
                            КонецЕсли;
                        КонецЕсли;
                                                Прервать;
                    КонецЕсли;
                КонецЕсли;
            КонецЦикла;
        КонецЦикла;
        Если m > 0 Тогда
                // расчет коэффициента ----------------------------
            t = Цел(tr/2);
            dj = (m/a + m/b + (m-t)/m)/3;
            Если dj >= bt Тогда
                dw = dj + (L * p * (1-dj));
        Иначе
                dw = dj;
            КонецЕсли;
        КонецЕсли;
        // процент --------
        Если Процент Тогда
            dw = dw * 100;
        КонецЕсли;
        // результат
        Возврат dw;
КонецФункции /
```

Рисунок 5 – Программный код функции, реализующей алгоритм Джаро-Винклера

Следует отметить, что помимо точного соответствия наименований, программа выявляет элементы «похожие», «мало похожие», «не похожие», а также отсутствующие в одной из баз данных.

3.2 Региональный мастер-индекс пациентов на платформе InterSystems HealthShare

Предлагаемое ИТ-решение предназначено для повышения функциональных возможностей Единой государственной информационной системы в сфере здравоохранения (ЕГИСЗ).

В системе управления здравоохранением существует проблема разрозненности медицинской информации (электронные медицинские карты, архивы изображений и др.) в региональных информационных системах, аккумулирующих данные из разнородных источников.

Для решения данной проблемы в ЕГИСЗ внедряется Региональный мастер-индекс пациентов (РМП) на платформе InterSystems HealthShare, который обеспечивает однозначную идентификацию пациентов, в том числе по неточным или неполным демографическим данным благодаря применению вероятностных моделей и алгоритмов нестрогого соответствия [18].

Архитектура регионального сегмента ЕГИСЗ с РМП представлена на рисунке 6.

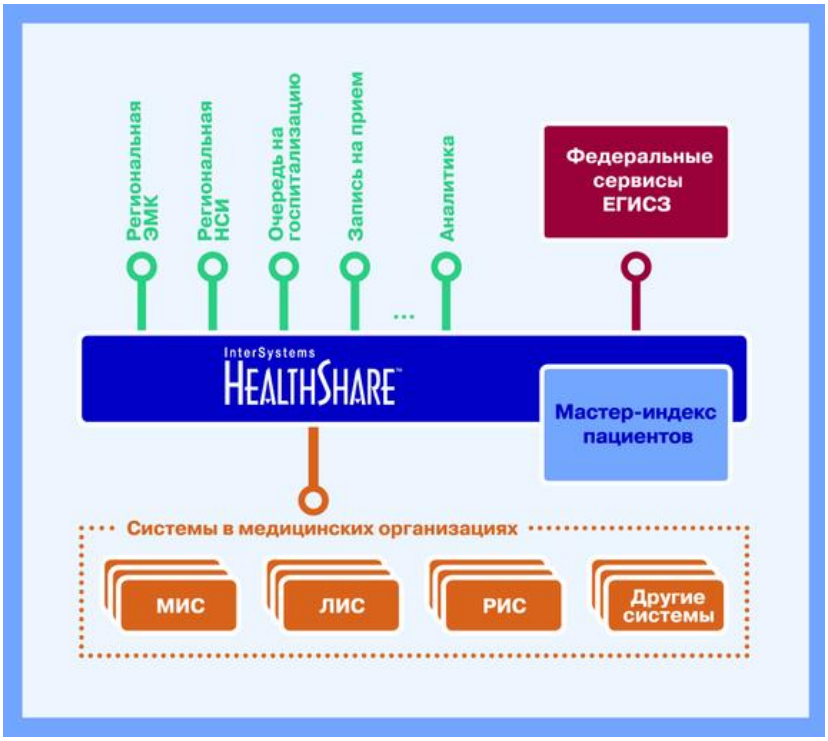

Рисунок 6 - Архитектура регионального сегмента ЕГИСЗ с РМП

При вычислении вероятности совпадения используются весовые коэффициенты, метрики, фонетические алгоритмы и правила нормализации, заранее настроенные для ключевых реквизитов, таких как Ф.И.О., дата рождения, пол, адрес и т.п. (рисунок 7).

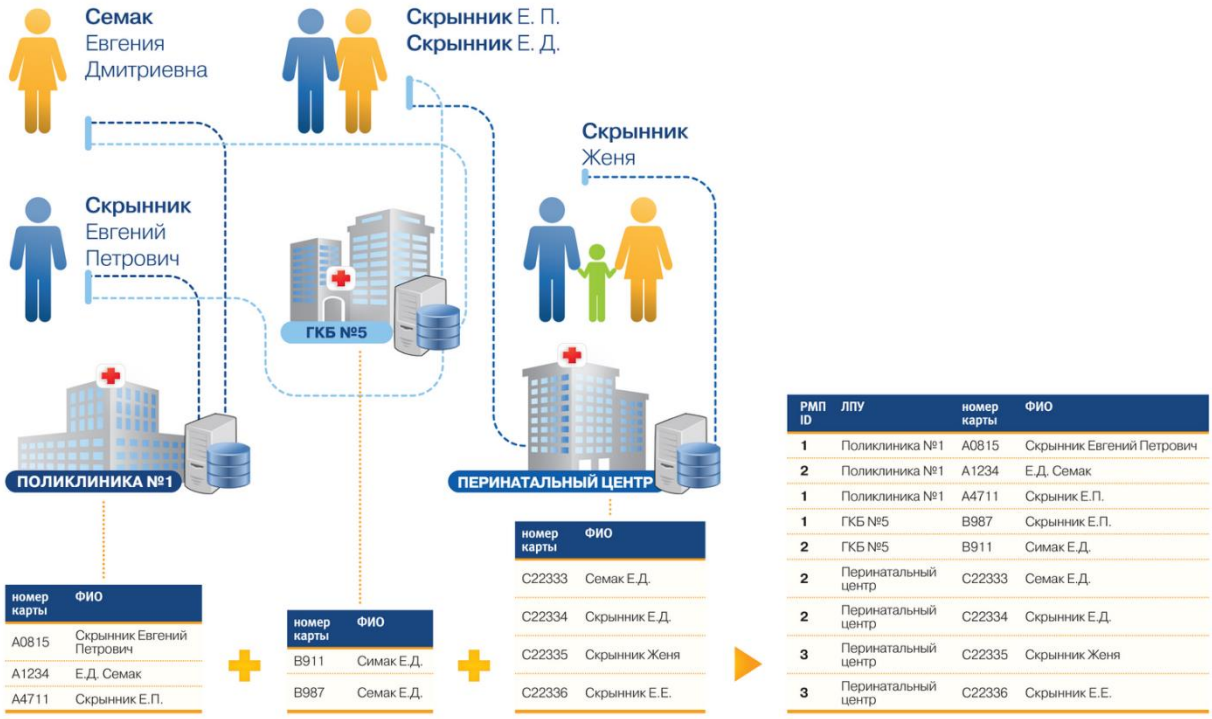

Рисунок 7 – Пример работы алгоритмов РМП

Классифицирование пар записей:

– вес пары записей равен сумме весов, полученных при сравнении;
– ключевых реквизитов с учетом их специфики с использованием;
– алгоритмов нестрогого сопоставления;
– фонетических алгоритмов;
– частотных словарей.

В рамках HealthShare реализован алгоритм Вагнера-Фишера. Стоимость одной операции любого типа принимается за единицу.

Как показывает практика, РМП обеспечит экономию средств ОМС за счет сокращения количества повторных исследований, уже проведенных для

данного пациента в других медицинских организациях, а также позволит повысить качество медицинской помощи благодаря снижению числа ошибочных назначений.

3.3 Информационная система «Фольклор»

Информационная система (ИС) «Фольклор» представляет собой лингвистическую программу обработки текстов на естественном языке, разработанную на основе теоретико-графовых моделей [5].

Структурная схема ИС приведена на рисунке 8.

Рисунок 8 – Структурная схема ИС «Фольклор»

Тексты хранятся в файлах формата TextGML. В каждом файле содержится отдельный текст коллекции, его характеристики и соответствующая теоретико-графовая модель.

Морфологический анализ текстов организован следующим образом: ИС выделяет в тексте отдельные слова, а затем осуществляет их поиск в морфологическом словаре.

Если слово отсутствует, то пользователю предлагается несколько схожих по написанию слов, к которым данное слово может быть отнесено в качестве словоформы.

Схожесть слов определяется при помощи алгоритма сравнения строк Вагнера-Фишера.

Следует также отметить, что при определении расстояния на множестве графов в ИС также используется алгоритм Вагнера-Фишера. Расстояние ищется как последовательность операций редактирования (вставка, удаление и переименование вершин и ребер), которая преобразует один граф в другой с минимальным суммарным весом.

ИС разработана в среде программирования Delphi 7.0.

3.4 Разработка программы поиска клиента в перечне Росфинмониторинга

На практике часто возникает задача поиска конкретной записи в заданном форматированном списке по составному ключу, представляющему собой определенный набор атрибутов.

Например, в финансовых организациях (банках, страховых компаниях, микрофинансовых структурах и т.д.) при заключении договора об оказании финансовых услуг необходимо предварительно проверить клиента на предмет наличия в Перечне организаций и физических лиц Федеральной службы по финансовому мониторингу (Росфинмониторинг), в отношении которых имеются сведения об их причастности к экстремистской деятельности или терроризму [13].

Ключ поиска представляет собой совокупность таких атрибутов клиента, как фамилия, имя, отчество, дата рождения, адрес места жительства и др.

Для повышения качества финансовых услуг и обеспечения безопасности финансовых операций очень важно не только найти заданный элемент в списке, но и сделать это достаточно быстро.

На рисунке 9 приведен пример записи элемента Перечня организаций и физических лиц Федеральной службы по финансовому мониторингу.

ФИЗИЧЕСКИЕ ЛИЦА				
№ п/п	Фамилия, имя, отчество	Дата и место рождения	Данные документа, удостоверяющего личность	Адрес места жительства или места нахождения
1	2	3	4	5
1.	ИВАНОВ ИВАН ИВАНОВИЧ*	19.12.1986 г., с. Рябиновое	Паспорт РФ: выдан ОВД	

Рисунок 9 - Пример записи элемента перечня

Перечень доступен для скачивания с личного кабинета юридического лица в форматах DBF, DOCX, XML.

Задача заключается в поиске элемента по заданному ключу в исходном списке.

Ключ представляет собой множество:

$$K = (A_1, A_2, \ldots, A_n),$$

где:

A_1, A_2, …, A_n – атрибуты элемента, обеспечивающие его однозначную идентификацию (например, фамилия, имя, отчество, дата рождения, серия и номер паспорта или водительского удостоверения).

В страховой ИС «Континент-Страхование 8» для решения данной задачи используется линейный поиск, причем для сравнения используются стандартные строковые функции языка 1С8 [8].

Как показала практика, процесс поиска в данной системе занимает несколько часов, причем выявляются только записи, полностью совпадающие с шаблоном.

Это снижает эффективность поиска, т.к. не позволяет учитывать возможные ошибки операциониста при вводе данных клиента в базу данных (БД) ИС.

Для решения данной проблемы предлагается использовать методику, которая состоит из следующих действий:

1) нормализация строки – это процесс преобразования строки в единую каноническую форму, которую она могла бы не иметь раньше.

Нормализация строки предназначена для повышения производительности процедуры ее обработки и сравнения.

Характер нормализации зависит от синтаксических особенностей языка, на котором составлена строка.

Следует учесть, что нет универсальной процедуры нормализации.

Наиболее эффективным способом нормализации строк является удаление в них промежуточных пробелов и перевод всех ее символов в верхний регистр;

2) индексирование и сортировка списка.

Механизм индексирования широко применяется в системах управления базами данных (СУБД).

Индекс БД представляет собой структуру данных, которая увеличивает скорость операций поиска данных в таблице БД за счет дополнительных объемов записи и хранения для поддержания структуры данных индекса.

Индексы используются для быстрого поиска данных без необходимости исследования каждой строки в таблице БД при каждом обращении к ней.

Индексы могут быть созданы с использованием одного или нескольких столбцов таблицы БД, что обеспечивает основу для быстрого случайного поиска и эффективного доступа к упорядоченным записям.

Некоторые базы данных расширяют возможности индексации, позволяя разработчикам создавать индексы для функций или выражений.

Таким образом, для применения механизма индексирования к конкретному списку необходимо преобразовать его в таблицу формата, доступного для обработки конкретной БД;

3) поиск строки с реквизитами клиента из БД в Перечне с помощью алгоритма Джаро-Винклера (рисунок 10).

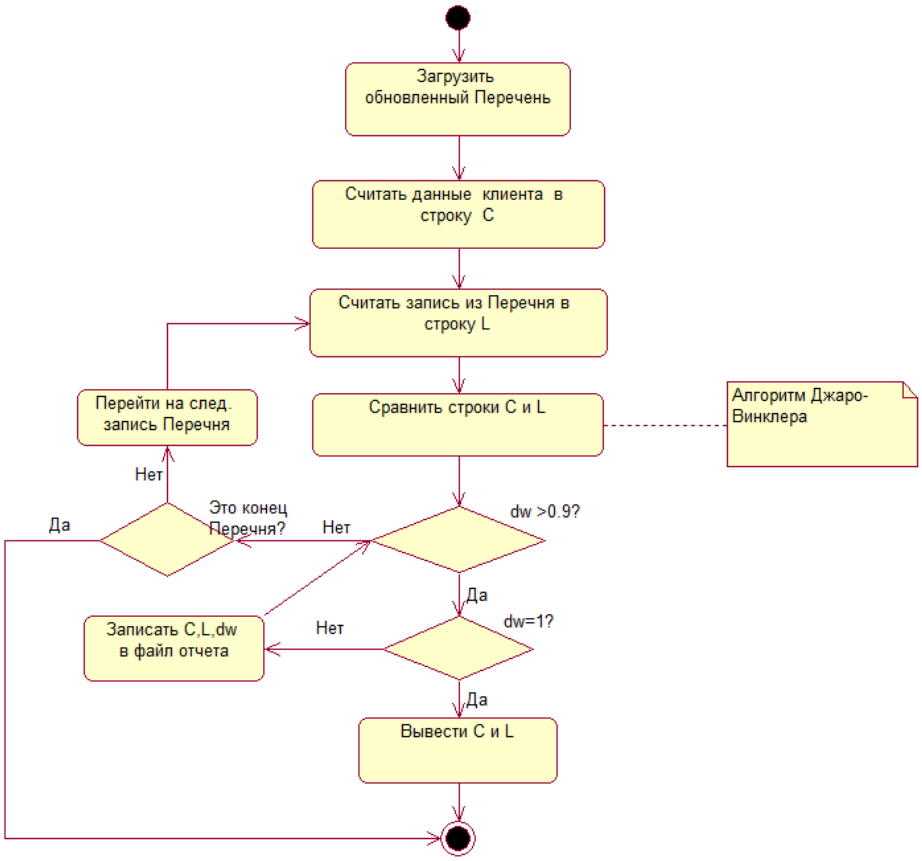

Рисунок 10 – Диаграмма деятельности алгоритма поиска клиента в Перечне Росфинмониторинга

Как следует из алгоритма, для дальнейшей проверки используются совпадения с $d_w > 0.9$.

Приложение, выполняющее поиск по представленному алгоритму, реализовано в среде VB.NET + СУБД MS SQL Server 2012.

Для повышения производительности алгоритм Джаро-Винклера реализован в виде функции на языке T-SQL (рисунок 11).

```
USE [INSURANCEDB]
GO
/****** Object:  UserDefinedFunction [dbo].[ufn_JaroWinkler]    Script Date: 05/17/2020
08:20:20 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER FUNCTION [dbo].[ufn_JaroWinkler]
(
	-- Add the parameters for the function here
	@s1 nvarchar(100),
	@s2 nvarchar(100)
)
RETURNS numeric(11,10)
AS
BEGIN
	DECLARE @Result numeric(11,10)

	declare @prmKeyword as varchar(100), @prmCompareTo as varchar(100)
	declare @iProximity As int -- set the number of adjacent characters to compare to
	declare @i As int
	declare @x As int
	declare @iFrom As int
	declare @iTo As int
	declare @iMatchCharacters As int
	declare @iTransposeCount As int
	declare @iJaro As numeric(20, 10)
	declare @JaroWinkler As numeric(20, 10)

if len(@s1) <> 0 and len(@s2) <> 0

begin
	if len(@s1) > len(@s2)
	begin
		set @prmKeyword = @s1
		set @prmCompareTo = @s2
	end
	else
	begin
		set @prmKeyword = @s2
		set @prmCompareTo = @s1
	end
end

set @iProximity = 0
set @i = 0
set @x = 0
set @iFrom = 0
set @iTo = 0
set @iMatchCharacters = 0
```

```
set @iTransposeCount = 0
set @iJaro = 0
set @JaroWinkler = 0
set @prmCompareTo = UPPER(lTrim(rtrim(@prmCompareTo)))
set @prmKeyword = UPPER(lTrim(rtrim(@prmKeyword)))
If @prmCompareTo <> @prmKeyword
begin -- check if the two words are the same
	If charindex(@prmCompareTo, @prmKeyword) <= 0
	begin
		set @iProximity = (Len(@prmKeyword)/ 2) - 1-- compute for the proximity of
character checking allows matching characters to be up to X number of characters away.
		set @i = 1
		while @i <= Len(@prmKeyword)
		begin -- this is the index of the character to be compared to
			set @iTo = (@i + @iProximity) - 1
			-- get the left most side character based on the @iProximity
			If @i <= @iProximity
				set @iFrom = 1
			Else
				set @iFrom = @i - @iProximity + 1

			-- start the letter by letter comparison
			set @x = @iFrom
			while @x <= @iTo
			begin
				If Substring(@prmKeyword, @i, 1) = Substring(@prmCompareTo,
@x, 1)
				begin
					If @i = @x
					begin
						set @iMatchCharacters = @iMatchCharacters + 1
						break
					End
					set @iMatchCharacters = @iMatchCharacters + 1
					set @iTransposeCount = @iTransposeCount + 1
					break
				End
			set @x = @x +1
			Continue
			end
		set @i = @i +1
		Continue
		end
		set @iTransposeCount = cast(@iTransposeCount / 2 as int)
		If @iMatchCharacters > 0
		begin
			set @x = 0
			set @i = 1
			while @i <= 4
			begin
				If Substring(@prmKeyword, @i, 1) = Substring(@prmCompareTo,
@i, 1)
```

```
                    set @x = @x + 1
                Else
                    break
            set @i = @i +1
            Continue
            end
            set @iJaro = ((cast(@iMatchCharacters as numeric(20,10)) /
cast(Len(@prmKeyword) as numeric(20,10)))
                    + (cast(@iMatchCharacters as numeric(20,10)) /
cast(Len(@prmCompareTo) as numeric(20,10)))
                    +((cast(@iMatchCharacters as numeric(20,10)) -
cast(@iTransposeCount as numeric(20,10)))
                    / cast(@iMatchCharacters as numeric(20,10)))) / 3
            If @x > 0
                set @JaroWinkler = @iJaro + 0.1 * @x * (1 - @iJaro)
            Else
                set @JaroWinkler = @iJaro
        end
        Else
            set @JaroWinkler = 0
    end
    Else -- return 1 result if the keyword is within the search string
        set @JaroWinkler = 1
end
Else -- return a 1 result if the string are the same
    set @JaroWinkler = 1
    set @result = @JaroWinkler
    RETURN @Result
END
```

Рисунок 11 – Программный код T-SQL функции Джаро-Винклера.

Указанная функция вызывается из хранимой процедуры СУБД:

```
USE [INSURANCEDB]
GO
/****** Object:  StoredProcedure [dbo].[sp_al_dw]    Script Date:
05/17/2020 14:22:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[sp_al_dw]
@var1 nvarchar(100)
AS
```

```
BEGIN
    SET NOCOUNT ON;
SELECT dbo.ufn_JaroWinkler(@var1,Perechen_rfm.Person) from
Perechen_rfm
END
```

На рисунке 12 изображена диаграмма классов программы, отражающей ее статический аспект [10].

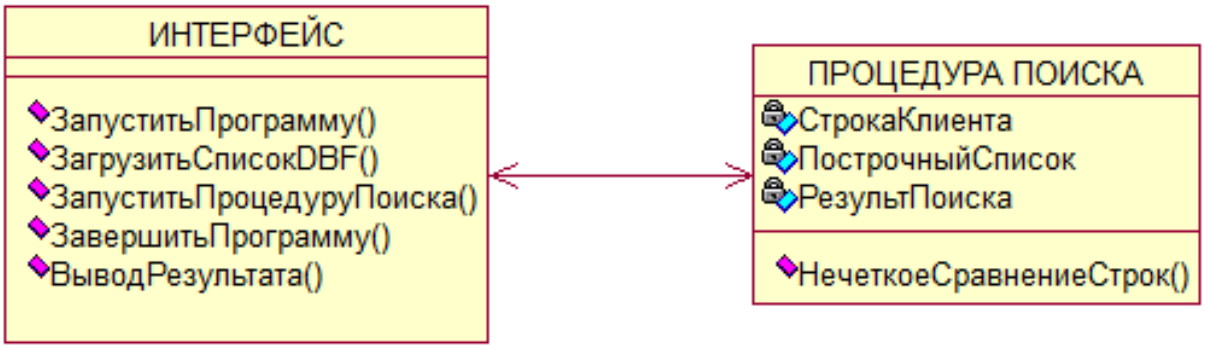

Рисунок 12 – Диаграмма классов программы

На рисунках 13, 14 представлены окно выполнения программы и отчет результатов тестирования алгоритма поиска в виде файл Excel, соответственно.

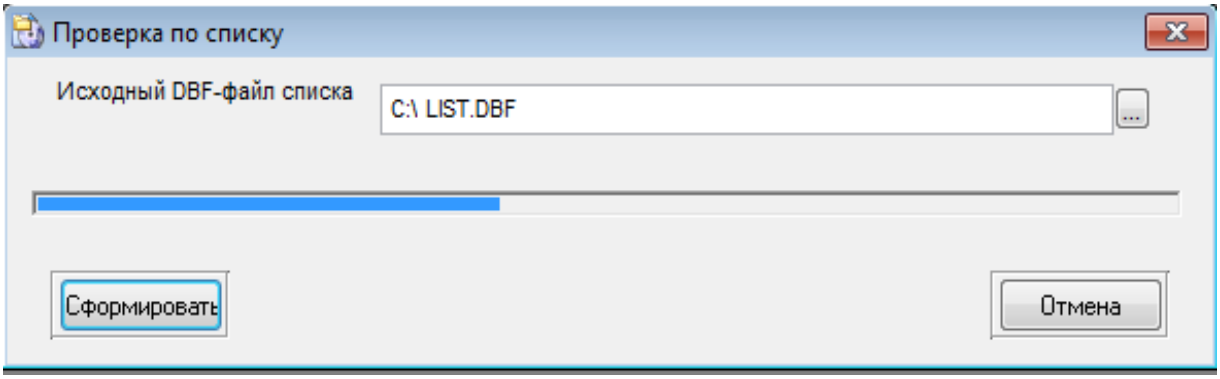

Рисунок 13 – Окно выполнения программы для сравнения алгоритмов поиска элементов в списке

Шаблон	Элемент списка	dw
Петров Петр Петрович	Иваненко И.Л.	0,42
Петров Петр Петрович	Иванов И.И.	0,44
Петров Петр Петрович	Иванов И И	0,45
Петров Петр Петрович	Иванов Иван И.	0,47
Иванов Иван Иванович	Петров П.П.	0,57
Иванов Иван Иванович	Петровский П.	0,62
Иванов Иван Иванович	Петрович П.П.	0,62
Петров Петр Петрович	Иванов Иван Иванович	0,68
Иванов Иван Иванович	Иваненко И.Л.	0,87
Петров Петр Петрович	Петровский П.	0,88
Петров Петр Петрович	Петрович П.П.	0,88
Петров Петр Петрович	Петров П.П.	0,92
Иванов Иван Иванович	Иванов И.И.	0,94
Иванов Иван Иванович	Иванов Иван И.	0,95
Иванов Иван Иванович	Иванов И И	0,96
Петров Петр Петрович	Петров Петр Петрович	1,00
Иванов Иван Иванович	Иванов Иван Иванович	1,00

Рисунок 14 – Результаты тестирования алгоритма (красным цветом выделены строки, используемые для дальнейшего анализа)

Исходя из результатов тестирования программы, представленных на рисунке 14, применение алгоритма Джаро-Винклера для поиска клиентов в Перечне Росфинмониторинга позволило повысить функциональные возможности данной программы.

Кроме того, применение механизмов оптимизации позволило увеличить производительность поиска по сравнению с типовой обработкой страховой ИС «Континент: Страхование 8» почти в 20 раз.

Выводы к главе 3

1. Как показывает практика, алгоритмы нечеткого сравнения строк применяются для таких прикладных задач, как сравнение наименований элементов справочников двух баз данных 1С8, однозначная идентификация пациентов медучреждений, в том числе по неточным или неполным

демографическим данным, морфологический анализ текстов на естественных языках и др.

2. Применение алгоритма Джаро-Винклера для нечеткого сравнения и механизмов оптимизации линейного алгоритма в программе поиска клиентов в Перечне Росфинмонитринга позволило повысить функциональные возможности данной программы и увеличить быстродействие поиска почти в 20 раз.

3. При написании программы, был задействован механизм индексирования, который применяется в системах управления базами данных (СУБД), что позволило существенно увеличить скорость операций поиска данных в таблице БД.

4. В настоящее время, данные алгоритмы частично задействованы в таких продуктах, как Elasticsearch – современная поисковая система, в которой поиск происходит с учетом морфологии языка или поиск по geo координатам.

Заключение

Бакалаврская работа посвящена актуальной проблеме анализа методов и алгоритмов нечеткого сравнения строк и исследования их применения при разработке программного обеспечения для решения различных прикладных задач.

В ходе выполнения бакалаврской работы достигнуты следующие результаты:

1. Проанализированы методы нечеткого сравнения строк.

Как показал анализ, наиболее популярными считаются методы нечеткого сравнения строк, основанные на символьном подходе. К ним относятся методы Дамерау-Левенштейна и Джаро-Винклера. С помощью Левенштейна можно подсчитать количество операций, которые необходимы для того, чтобы преобразовать одну строку в другую. Метод Дамерау-Левенштейна также включает транспонирование в ряд отдельных операций. С помощью расстояния Джаро измеряется сходство между двумя строками. Метод Джаро-Винклера позволяет дать более благоприятные оценки для строк, которые соответствуют заданной длине префикса в начале строки.

2. Проанализированы популярные алгоритмы нечеткого сравнения строк.

Как показал анализ, все алгоритмы нечеткого сравнения строк обладают определенными достоинствами и недостатками. Выбор того или иного алгоритма зависит от конкретной прикладной задачи и требований по ее программной реализации.

3. Исследованы области применения алгоритмов нечеткого сравнения строк для решения различных задач.

Как показывает практика, алгоритмы нечеткого сравнения строк применяются для таких прикладных задач, как сравнение наименований элементов справочников двух баз данных 1С8, однозначная идентификация

пациентов медучреждений, в том числе по неточным или неполным демографическим данным, морфологический анализ текстов на естественных языках и др.

4. Разработана программа поиска клиентов в Перечне Росфинмониторинга.

Применение алгоритма Джаро-Винклера для нечеткого сравнения и механизмов оптимизации линейного алгоритма в программе поиска клиентов в Перечне Росфинмониторинга позволило повысить функциональные возможности данной программы и увеличить быстродействие поиска почти в 20 раз.

Результаты бакалаврской работы представляют практический интерес и могут быть рекомендованы для разработчиков программного обеспечения, основанного на применении методов нечеткого сравнения строк.

Список используемой литературы и используемых источников

1. ГОСТ 19.402–78. Единая система программной документации. Описание программы. URL: http://www.standards.ru/document/4153976.aspx (дата обращения: 30.04.2020).

2. ГОСТ 28806-90. Качество программных средств. Термины и определения. URL: http://www.standards.ru/document/4122406.aspx (дата обращения: 30.04.2020).

3. Левенштейн В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов // Докл. АН СССР. 1965. 163 (4). С. 845–848.

4. Лещенко А.В. Практическое применение алгоритмов нечеткого поиска // Сборник научных трудов НГТУ. 2018. №3-4(93). С. 59-69.

5. Москин Н. Д. Инструменты исследования текстовых коллекций на основе теоретико-графовых моделей в информационной системе «Фольклор» // Прикладная информатика. 2010. №4. С. 48-62.

6. Орлов Д. Подсистема сопоставления записей в хранилище данных [Электронный ресурс]. URL:https://www.sql.ru/articles/datawarehousing/datawarehouse_record_linkage.shtml (дата обращения: 30.04.2020).

7. Поиск нечетких соответствий: сравнение записей по расстоянию между строками [Электронный ресурс]. URL: https://www.megaputer.com/ru/fuzzy-matching-comparing-records-with-string-distance-measures/ (дата обращения: 30.04.2020).

8. Программный продукт «Континент: Страхование 8» [Электронный ресурс]. URL: https://kontinent.systems/ (дата обращения: 30.04.2020).

9. Решетников А.Д. О подходах для определения меры несходства в текстовых данных // Вестник Воронежского института высоких технологий. №3(30). 2019. С. 35-39.

10. Самуйлов С.В. Объектно-ориентированное моделирование на основе UML [Электронный ресурс]: учебное пособие. Саратов: Вузовское образование. 2016. 37 с. URL: http://www.iprbookshop.ru/47277.html (дата обращения: 30.04.2020).

11. Сверка справочников между базами 1С v8.1 [Электронный ресурс]. URL: https://infostart.ru/public/80077/ (дата обращения: 30.04.2020).

12. Сходство Джаро - Винклера. Нечеткое сравнение строк [Электронный ресурс]. URL: http://xn---1-6kca8bgsjrjhe.xn--p1ai/public/1172479/ (дата обращения: 30.04.2020).

13. Федеральная служба по финансовому мониторингу [Электронный ресурс]. URL: http://www.fedsfm.ru/ (дата обращения: 30.04.2020).

14. Brinardi L., Seng H. Text Documents Plagiarism Detection using Rabin-Karpand Jaro-Winkler Distance Algorithms, Indonesian J. of Electrical Engineering and Computer Science, 2017, v.5(2), P. 462-471.

15. Code Spelunking: Jaro-Winkler String Comparison [Электронный ресурс]. URL: https://lingpipe-blog.com/2006/12/13/code-spelunking-jaro-winkler-string-comparison/ (дата обращения: 30.04.2020).

16. Damerau-Levenshtein distance [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein_distance(дата обращения: 30.04.2020).

17. Damerau-Levenshtein Edit Distance Explained [Электронный ресурс]. URL:https://www.lemoda.net/text-fuzzy/damerau-levenshtein/ (дата обращения: 30.04.2020).

18. InterSystems: Региональный мастер-индекс пациентов [Электронный ресурс]. URL:http://zdrav.expert/index.php/%D0%9F%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82:InterSystems:_%D0%A0%D0%B5%D0%B3%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%82%D0%B5%D1%80-%D0%B8%D0%BD%D0%B4%D0%B5%D0%BA%D1%81_%D0%BF%D0%B0

%D1%86%D0%B8%D0%B5%D0%BD%D1%82%D0%BE%D0%B2_(%D0%A0%D0%9C%D0%9F) (дата обращения: 30.04.2020).

19. Jaro distance [Электронный ресурс]. URL: https://rosettacode.org/wiki/Jaro_distance(дата обращения: 30.04.2020).

20. Natural Language Processing for Fuzzy String Matching with Python [Электронный ресурс]. URL: https://towardsdatascience.com/natural-language-processing-for-fuzzy-string-matching-with-python-6632b7824c49 (дата обращения: 30.04.2020).

21. Navarro G. A guided tour to approximate string matching, ACM Computing Surveys, 2001, v.33 (1), P. 31–88.

22. Vijay Nath, Jyotsna Kumar Mandal (Eds). Proceedings of the Third International Conference on Microelectronics, Computing and Communication Systems: MCCS 2018, Singapore : Springer, Lecture Notes in Electrical Engineering, 2019, v.556, 669 p.

23. Wagner R. A., Fischer M. J. The string-to-string correction problem. J. ACM, v. 21(1), 1974. P. 168.

24. Wagner–Fischer algorithm [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Wagner%E2%80%93Fischer_algorithm (дата обращения: 30.04.2020).

25. Winkler W. E. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage, Proceedings of the Section on Survey Research Methods, American Statistical Association, 1990, P. 354–359.