Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

# СОВМЕЩЕНИЕ АЛГОРИТМОВ НЕЧЕТКОГО ПОИСКА И ФОНЕТИЧЕСКИХ АЛГОРИТМОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Ибрагимовой Мариам Анвяровны

Научный руководитель
доцент, к. ф.-м. н. _____ И. А. Батраева

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2016

# ВВЕДЕНИЕ

Современным информационным системам присущ рост количества хранимой и обрабатываемой информации. Большую часть этой информации составляет информация, потребляемая человеком — медиаконтент. Эта информация разнородна — основными её видами являются текст, аудиофайлы и видеофайлы. Поиск текстовой информации является наиболее востребованным, так как она составляет наибольшую её часть и используется во всех сферах человеческой деятельности.

Для поиска по наборам текстовой информации создаются поисковые системы. Эти системы позволяют отбирать тексты, соответствующие (релевантные) определённому запросу. В общем случае, запрос представляет собой строку, которая, предположительно, содержится в искомом документе. На этом этапе возникают сложности, связанные с человеческим фактором — запрос может содержать разного рода ошибки, которые могут совершаться по случайности, либо вследствие неграмотности. Также возможна ситуация, когда запрос составляется «на слух», без знания о том, как его правильно записать. Для преодоления этих препятствий существуют алгоритмы, применяемые для так называемого нечёткого поиска. Задачей таких алгоритмов является нахождение элементов, которые наиболее близки по написанию и/или произношению запросу.

Близость строк по написанию определяется с помощью алгоритмов, использующих некоторую метрику — функцию расстояния, которая сопоставляет двум строкам некоторое число, по которому можно судить об их различии. В числе наиболее известных метрик — расстояния Хемминга, Левенштейна и Дамерау-Левенштейна. Как правило, такие алгоритмы не берут во внимание языки запроса и множества элементов, по которым ведётся поиск — для них важна только операция сравнения символов.

Суть фонетических алгоритмов — группировка слов со схожим произношением с помощью строки-кода, которая из них извлекается. Таким образом, код объединяет несколько слов. Существует множество алгоритмов фонетического кодирования — они отличаются сложностью реализации, скоростью работы и языком применения, ведь каждый язык имеет свою фонетику. Большинство алгоритмов предназначены для английского языка, но многие из них можно адаптировать для других, в том числе, и для русского.

В данной дипломной работе рассматривается совместное применение алгоритмов нечеткого поиска и фонетических алгоритмов. Комбинирование этих двух видов алгоритмов может оказаться более эффективным с точки зрения точности и релевантности результатов, чем использование только метрик, так как последние не берут во внимание фонетику. Более того, использование только метрики сопряжено с высокими расходами процессорного времени — в тривиальном случае для поиска вычисляется расстояние между запросом и словами из словаря, количество которых может исчисляться десятками тысяч, а вычислительная сложность метрики в лучшем случае — $O(N + M)$, где $N$ и $M$ — длины строк, для которых вычисляется расстояние. С другой стороны, фонетические алгоритмы тоже обладают специфическими недостатками — простая опечатка, такая как пропуск буквы, может изменить код и без сравнения кода запроса с кодами элементов посредством метрики в результате может не оказаться ни одного из релевантных элементов.

Целью дипломной работы является программная реализация метрики Дамерау-Левенштейна, N-граммной метрики, фонетических алгоритмов Soundex и Metaphone, а также их сочетаний для исследования их свойств в контексте производительности и качества поиска.

Дипломная работа состоит из введения, четырех разделов, заключения, списка использованных источников и приложения.

В разделе 1 приводятся описание алгоритмов Дамерау-Левенштейна, N-грамм, Soundex и Metaphone, а также рассматриваются их параметры и характеристики.

В разделе 2 рассматриваются аспекты совместного использования описанных ранее алгоритмов.

В разделе 3 представлено описание и назначение разработанной программы, предназначенной для анализа алгоритмов и их совместного применения.

В разделе 4 приведены результаты проведенных экспериментов для иллюстрации характеристик алгоритмов и их комбинации.

Приложения А, Б, В и Г содержат фрагменты программного кода приложения — реализацию алгоритмов.

## 1 Основное содержание

В данной работе рассмотрены четыре алгоритма, два из которых являются алгоритмами нечеткого поиска, а другие два — фонетические алгоритмы.

Первый алгоритм, рассмотренный в работе и использованный в программе — алгоритм Дамерау-Левенштейн. Для того, чтобы описать работу данного алгоритма, следует охарактеризовать более простой аналог данного алгоритма — алгоритм Левенштейна [1].

Алгоритм Левенштейна (или «Дистанция Левенштейна») — один из наиболее популярных алгоритмов нахождения расстояния между двумя строками. Характеризуется минимальным количеством операций вставки, удаления и замены одного символа на другой, необходимых для преобразования одной строки в другую. Данный алгоритм позволяет наглядно определить, насколько одна строка отличается от другой и за какое количество шагов одна строка может быть преобразована в другую [2].

Алгоритм Дамерау-Левенштейна, названный в честь Владимира Левенштейна и Фредерика Дамерау, отличается тем, что наряду с другими операциями в нём присутствует операция перестановки. В силу этого нахождение метрики становится более эффективным, так как большое количество ошибок при наборе текста составляют перестановки соседствующих символов. Наибольшую популярность данный алгоритм получил при разработке spell checker — программ для проверки орфографии, которые сейчас используются в большинстве сотовых телефонов [3].

Для получения расстояния между двумя строками $a$ и $b$ с соответствующими длинами $N$ и $M$ (индексация начинается с 0), а также определения редакционного предписания, необходимо построить матрицу расстояний D размерности $(N+1)*(M+1)$. Каждый элемент $D(i,j)$ содержит дистанцию между первыми $i$ символами строки $a$ и первыми $j$ символами строки $b$. Строки матрицы соответствуют подстрокам $a$, а столбцы — подстрокам $b$. Строка или столбец с нулевыми индексами соответствуют пустой подстроке. Элемент $D(N+1, M+1)$ содержит искомое расстояние между строками $a$ и $b$, все остальные элементы содержат расстояние между подстроками соответствующих их индексов [4].

Также часто требуется нахождение редакционного предписания — последовательности действий, необходимых для получения из первой строки второй

кратчайшим образом. Обычно действия обозначаются так: D (англ. delete) — удалить, I (англ. insert) — вставить, R (replace) — заменить, M (match) — совпадение, T (transposition) — транспозиция [5].

Вторым методом, использованным в приложении, является метод N-грамм. Этот метод — один из наиболее популярных методов, используемых при поиске слов. *N-граммой* называется последовательность из N элементов — букв, звуков, слогов или даже слов. Суть метода заключается в хранении соответствий между N-граммами и словами, в которых данная N-грамма встречается. В данной работе применяются триграммы — последовательности из трех букв. Как показывает опыт, это число является наиболее оптимальным — диграммы *(N = 2)* создают слишком большие множества слов, которые их включают, т.е. диграмма плохо характеризует слово. При *N* более четырех возникает ограничение на длину слова. Дополнительно можно определить *N-граммное расстояние* как отношение количества общих для двух слов N-грамм, к их общему количеству в этих словах [6]. Основным недостатком данного алгоритма является неустойчивость к некоторого рода ошибкам. Так, при поиске строки «вота» ни по одной из триграмм исходного слова — вот ота— не будет найдено искомое «вода». Отсюда можно увидеть, что при работе данного алгоритма предпочтительней, если ошибка будет находится или в начале, или в конце слова, т.к. в таком случае подходящее слова не будет найдено лишь по одной триграмме. Также отсюда следует, что чем меньше длина слова и чем больше в нем ошибок, тем выше шанс того, что оно не попадет в соответствующие триграммам запроса списки, и не будет присутствовать в результате.

Третьим алгоритмом, использованным в программе для поиска слов, является алгоритм Soundex. Он относится к группе фонетических алгоритмов и производит индексацию слов по звучанию Применяется для слов английского языка. Разработан Робертом Расселом и Маргарет Кинг Оделл и запатентован в 1918 и 1922 г. Наибольшую популярность этот алгоритм получил после того, как был описан Дональдом Кнутом в одной из книг серии «Искусство программирования» [7]. Цель алгоритма состоит в том, чтобы омофоны были закодированы в том же представлении, и таким образом слова будут сопоставимы, несмотря на незначительные различия в орфографии.

Вычисляемый с помощью данного алгоритма код будет состоять из 4 символов — одной буквы(первой буквы слова) и трех следом идущих числа

(от 1 до 9). Недостатком данного алгоритма является то, что он используется только для английского языка, т.е. правила алгоритма не применимы к русскому языку. Однако, проанализировав правила русского языка и модифицировав оригинальный алгоритм, можно получить последовательность шагов и для русского языка. Для точности поиска была взята длина кода в 6 символов, и первая буква тоже кодируется [8].

Вторым фонетическим алгоритмом, который рассматривается в работе, является Metaphone — фонетический алгоритм, разработанный Лоуренсом Филипсом в 1990 году в качестве альтернативы алгоритму Soundex. Он преобразует исходное слово в код в соответствии с правилами английского языка, при этом используются более сложные правила — замена происходит не просто на основании букв, а буквосочетаний, благодаря чему получается более точный код. Полученные ключи имеют переменную длину. Аналогично схожим алгоритмам, одинаковые слова будут иметь одинаковый код. Полученный посредством алгоритма Metaphone код будет представлять собой набор символов из множества 0BFHJKLMNPRSTWXY, в начале слова также могут быть гласные из множества AEIOU [9].

В 2002 году была опубликована статья Петра Каньковски, в которой Metaphone был адаптирован к русскому языку. Этот алгоритм, называемый Русским Metaphone, учитывает правила и нормы русского языка при преобразовании исходного слова в код.

Выбор данных алоритмов сделан в связи с тем, что они являются одними из самых популярных алгоритмов в своих группах, а также за счет простоты реализации.

Идея их совмещения связана с тем, что их совместное использование позволяет взять лучшее в них, уменьшив влияние их недостатков. Так алгоритм Дамерау-Левенштейна имеет сложность *O(N*M)*, в связи с чем поиск по словарю среднего или большого размера будет осуществляться достаточно долго. Алгоритм Soundex сам по себе не так эффективен за счет того, что без использования дополнительных алгоритмов данный алгоритм уязвим перед многими видами опечаток. Например, пропуск буквы в начале слова приведет к тому, что коды слов будут совершенно различны. Подбными недостатками обладает и Metaphone. Метод N-грамм имеет плохую точность, что связано с большими списками слов, которые соответствуют одной триграмме.

Совмещение алгоритмов Дамерау-Левенштейна и Soundex осуществляется по следующей схеме:

1. Для каждого слова в словаре вычисляется его Soundex-код, и сохраняется в ассоциативном массиве (код сопоставляется множеству слов из словаря);
2. Поиск работает по принципу двойного отсеивания: для каждого кода-ключа в ассоциативном массиве вычисляется расстояние Дамерау-Левенштейна с кодом запроса и отдельным образом сохраняется. После этого, из всех кодов отбираются те, для которых вычисленное расстояние не выше некоторого порога (в данном случае — половине длины кода, что при заданных стоимостях равно замене двух символов или двум транспозициям).
3. Для всех слов, соответствующих отобранным кодам вычисляется расстояние Дамерау-Левенштейна, по которому и формируется список результатов.

Что касается второй пары алгоритмов — Metaphone и метода N-грамм — то их совмещение основывается на схожем принципе:

1. Для каждого слова в словаре вычисляется его Metaphone-код, разбивается на триграммы, которые сохраняются в ассоциативном массиве (триграмма сопоставляется множеству индексов слов из словаря, код которых такую триграмму содержит). Также вычисляется Metaphone-код запроса;
2. Поиск работает по принципу двойного отсеивания: Metaphone-код исходного слова разбивается на триграммы, и из ассоциативного массива собираются индексы слов, соответствующие им;
3. Исходное слово разбивается на триграммы, и из ассоциативного массива собираются индексы слов, в которые данная триграмма входит. Затем, для отобранных слов и запроса вычисляется триграммное расстояние, по которому и формируется список результатов.

Таким образом, совместное использование алгоритмов сохраняет большинство ожидаемых ответов, а благодаря применению алгоритмов Дамерау-Левенштейна и N-грамм лишь на части всего словаря поиск осуществляется значительно быстрее.

Программа, предназначенная для анализа алгоритмов Дамерау-Левен-

штейна, N-грамм, Soundex и Metaphone, а также алгоритма, полученного при их совмещении, была реализована на языке программирования Java на платформе Java 8 с использованием использование платформы для создания приложений JavaFX [10].

Программа позволяет осуществлять нечеткий поиск по введенному слову шестью способами: с помощью алгоритма Дамерау-Левенштейна; с помощью алгоритма Soundex; с помощью совмещения алгоритмов Soundex и Дамерау-Левенштейна; помощью алгоритма Metaphone; с помощью алгоритма N-грамм; с помощью совмещения алгоритмов Metaphone и N-грамм. Поиск осуществляется по словарю русского языка, состоящему из 90000 слов.

Исходный код программы представляет собой два пакета, *fuzzy* и *ui*. Первый пакет включает в себя классы, имеющие непосредственное отношение к процессу поиска. Второй пакет содержит классы и файлы, необходимые для создания графического интерфейса. Пакет *fuzzy* содержит 6 классов: *Engine*, *DamerauLevenstein*, *Soundex*, *Trigram*, *Metaphone* и *SearchResult*. Пакет *ui* содержит классы *Controller* и *Main*, а также специальный файл *main.fxml*, использующийся в технологии JavaFX для создания графического интерфейса.

Программа работает следующим образом: при запуске приложения появляется окно, в верхней части которого присутствует поле для ввода слова, по которому и будет осуществлен поиск по словарю. Ниже присутствуют две радиокнопки с названиями «Расстояние Дамерау-Левенштейна и Soundex» и «Метод триграмм и Metaphone». При нажатии на одну из кнопок осуществляется выбор одной из пар алгоритмов, которые будут принимать участие в поиске. Ниже присутствуют два элемента CheckBox с названиями «Расстояние Дамерау-Левенштейн» и «Soundex». В зависимости от выбора — либо один из алгоритмов, либо они оба — и будут заданы соответствующие параметры поиска. После осуществления выбора и нажатия на кнопку «Поиск» в нижней части окна отображаются результаты поиска. Результатом поиска являются 10 максимально похожих слов, отображаемые сверху вниз от максимально похожего к менее похожим. В квадратных скобках справа от слов, являющимися результатами поиск, расположена дистанция между искомым словом и найденным. При поиске с использованием обоих алгоритмов в квадратных скобках указываются два числа: сначала дистанция по первом алгоритму, затем - по второму. Справа от результатов поиска расположена статистика поиска, в которой ука-

зано, за какое время был осуществлен поиск. При повторном осуществлении поиска новое время будет отображаться под значением последнего времени.

По умолчанию при запуске программы выбрана радиокнопка «Расстояние Дамерау-Левенштейна и Soundex», а также выбраны оба элемента CheckBox — «Расстояние Дамерау-Левенштейн» и «Soundex» (т.е. по умолчанию поиск осуществляется по совмещению алгоритмов Дамерау-Левенштейна и Soundex).

# ЗАКЛЮЧЕНИЕ

Основной целью работы являлось рассмотрение алгоритмов нечеткого поиска Дамерау-Левенштейна и N-грамм, фонетических алгоритмов Soundex и Metaphone, рассмотрение аспектов совместного использования алгоримов Soundex с Дамерау-Левенштейн и Metaphone с N-грамм, а также программная реализация алгоритмов по отдельности и совместно с анализом их эффективности. Результатами преддипломной работы являются следующие:

- рассмотрены алгоритмы нечеткого поиска Дамерау-Левенштейн и N-грамм;
- рассмотрены фонетические алгоритмы Soundex;
- адаптирован алгоритм Soundex для русского языка;
- адаптирован алгоритм Metaphone для русского языка;
- разработан вариант совместного использования алгоритмов Soundex и Дамерау-Левенштейн для русского языка;
- разработан вариант совместного использования алгоритмов Metaphone и N-грамм для русского языка;
- разработана программа, осуществляющая поиск по каждому из алгоритмов, а также по их совмещению;
- проведены численные эксперименты с разработанной программой и приведены соответствующие результаты.

Программная реализация показала, что совмещение алгоритмов обеспечивает более эффективный поиск за достаточно короткое время, в отличии от использования алгоритмов по отдельности. Полученная программа может быть адаптирована для поисковых системам.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1 *Wagner, R. A.* The String-to-String Correction Problem / R. A. Wagner. — Cambridge: Massachusetts Institute of Technology, 1974.

2 Задача о редакционном расстоянии, алгоритм Левенштейна [Электронный ресурс]. — URL: http://neerc.ifmo.ru/wiki/index.php/Levenshtein (Дата обращения 12.04.2016). Загл. с экр. Яз. рус.

3 Расстояние Дамерау-Левенштейн [Электронный ресурс]. — URL: http://elementy.ru/problems/1068/Rasstoyanie_DL (Дата обращения 22.04.2016). Загл. с экр. Яз. рус.

4 *Лиманов, Н.* Метод автоматизированного поиска персональных данных на основе нечеткого сравнения / Н. Лиманов. — Санкт-Петербург: Вестник, 2009.

5 *Маннинг, К.* Введение в информационный поиск / К. Маннинг. — Москва: Вильямс, 2014.

6 *Binstock, A.* Practical Algorithms for Programmers / A. Binstock. — Boston: Addison-Wesley, 1995.

7 *Кнут, Д.* Искусство программирования. Т. 3. Сортировка и поиск / Д. Кнут. — Москва: Вильямс, 2012.

8 Нечеткий поиск на клиенте и Soundex [Электронный ресурс]. — URL: https://habrahabr.ru/post/125617/ (Дата обращения 27.04.2016). Загл. с экр. Яз. рус.

9 *Jurafsky, D.* Speech and Language Processing / D. Jurafsky. — New Jersey: A Simon and Schuster Company, 2000.

10 *Машнин, Т.* JavaFX 2.0. Разработка RIA-приложений / Т. Машнин. — Санкт-Петербург: БХВ-Петербург, 2012.