Оглавление

[Введение 2](#_Toc73284457)

[Глава 1 Теоретическая часть 4](#_Toc73284458)

[1.1. Правила оформления таблиц 4](#_Toc73284459)

[1.2. Алгоритмы нечеткого поиска 7](#_Toc73284460)

[1.2.1. Расстояние Левенштейна 7](#_Toc73284461)

[1.2.1.1. Общее описание 7](#_Toc73284462)

[1.2.1.2. Алгоритм Вагнера-Фишера 7](#_Toc73284463)

[1.2.2. Алгоритм шинглов 9](#_Toc73284464)

[1.2.2.1. Канонизация текста 9](#_Toc73284465)

[1.2.2.2. Разбиение на шинглы 9](#_Toc73284466)

[1.2.2.3. Вычисление хэшей шинглов с помощью 84х статических функций 10](#_Toc73284467)

[1.2.2.4. Случайная выборка 84х значений контрольных сумм 10](#_Toc73284468)

[1.2.2.5. Сравнение, определение результата 10](#_Toc73284469)

[1.2.3. Алгоритм N-грамм 10](#_Toc73284470)

[1.3. Алгоритмы идентификации таблиц 11](#_Toc73284471)

[1.3.1. Rule-based алгоритмы 11](#_Toc73284472)

[1.3.1.1. TIRS 11](#_Toc73284473)

[1.3.1.2. Удаляй и властвуй 12](#_Toc73284474)

[1.3.2. Генетический алгоритм 14](#_Toc73284475)

[Список литературы 16](#_Toc73284476)

# Введение

Человек регулярно сталкивается с таблицами в повседневной жизни: расписание пар в университете, расписание кино, смета на работе. Персональный компьютер позволяет вывести работу с таблицами на новый уровень. Используя электронные таблицы, намного проще работать с данными, скорость расчётов повышается в разы. С применением еще и программирования, электронные таблицы позволяют решать очень сложные задачи.

Одной из самых популярных программ для работы с электронными таблицами является Excel. По данным сайта «excelwithbusiness» на данный момент Excel используют примерно 750 миллионов работников умственного труда [1]. Данная программа уже давно стал важной частью современного мира. Таблицы, в свою очередь, это неотъемлемая часть бизнеса, статистики, науки, домашнего хозяйства и т.д. Они могут варьироваться по сложности и использоваться по разным причинам, но основная их цель – это организация и категоризация данных в хорошо понимаемый для человека вид.

Как правило, составлением таблиц занимаются люди, организуют данные таким образом, как это удобно именно им [2]. При этом в большинстве случаев не учитываются возможность дальнейшего использования этих таблиц, например для автоматического извлечения данных или для интеграции. Это происходит как раз из-за низкого порога вхождения в Excel. В результате, когда таблица c непродуманной структурой попадает к программисту, ему сложно извлекать и анализировать ее. Ему приходится тратить много времени на понимание структуры, бороться со сложностями обработки таких данных.

В связи с вышеперечисленными проблемами, представляется значимым разработка алгоритмического и программного обеспечения, позволяющего проводить автоматизированную очистку табличных документов.

Для достижения этой цели был составлен список задач:

* Составить список требований к функциональности программного обеспечения;
* Определить наилучший стек технологий;
* Определить правила составления таблиц;
* На основе правил составления таблиц, определить алгоритмы для их реализации.

Для программной реализации представлены следующие требования:

* минимальные затраты от пользователя: ему нужно только выбрать нужный файл;
* автоматическое исправление орфографических ошибок;
* удаление или заполнение пустых ячеек;
* повышение качества структуры таблиц;
* приведение сокращений к единому стандарту;
* определение границ таблицы на листе

Введение раскрывает актуальность, объект, цель и задачи проекта, показывает практическую значимость работы.

В первой главе рассматриваются теоретические положения по составлению электронных таблиц и алгоритмы, которые могут помочь решить поставленную задачу.

Вторая глава посвящена программной части диплома, описанию достигнутых успехов.

В заключении подводятся итоги работы, оценивается качество написанного программного обеспечения, формируются итоговые выводы.

# Глава 1 Теоретическая часть

## Правила оформления таблиц

В соответствии с исследованиями, изложенными в статье «Data Organization in Spreadsheets» [3], для оформления таблиц следует придерживаться следующего списка правил.

1. Уницификация стандартов оформления данных в ячейках

Первое правило организации данных – быть последовательным. Это нужно для предотвращения траты времени на оптимизацию данных в будущем.

Рекомендуется использовать одинаковые названия для обозначения категорий. Для обозначения пола, например, нужно использовать «мужской» для мужчин, «женский» для женщин. Не нужно писать иногда «М, иногда «мужской», иногда «Мужской». Необходимо выбрать одно обозначение и использовать его.

Рекомендуется использовать фиксированные обозначения для отсутствующих значений. Лучше, чтобы каждая ячейка была заполнена, иначе не понятно, значение действительно отсутствует или его просто забыли вставить. Не нужно использовать числа для заполнения таких ячеек, это может сбить с толку в будущем. Также не нужно вставлять данные, объясняющие, почему значение отсутствует. Лучше выделить под это отдельный столбец.

Не рекомендуется изменять название переменной в файле. Если переменная названа «рост численности популяции за неделю», то дальше она везде должна писаться таким образом. Не нужно менять слова местами, добавлять новые и т.д.

Рекомендуется использовать один шаблон в разных файлаx. Если одна таблица лежит в нескольких файлах, то ее шаблон должен быть везде одинаковым.

Рекомендуется называть файлы похожим образом. Но не нужно использовать слово «финальный» в названии. Никогда нельзя знать, нужно будет переделывать файл в будущем или нет.

Необходимо быть аккуратным с дополнительными пробелами внутри ячейки. Пустая ячейка и ячейка с пробелами — это разные ячейки. Также же как «кот» и « кот ».

1. Канонические обозначения

Основные правила: не использовать пробелы, как в названиях переменных, так и в названиях файлов. Вместо них стоит использовать нижние подчеркивания или дефисы. Но не нужно использовать их одновременно, выберите что-то одно.

Не используйте специальные символы ($, @, %, #, & и т.д.) Использование этих символом может затруднить программную работу с вашими данными.

Главный принцип выбора имени – оно должно быть коротким, но значимым, поэтому не делайте его слишком коротким.

1. Не допускается оставлять пустые ячейки

Заполните все ячейки. Рекомендуется использовать одинаковое обозначение для отсутствующих данных. Лучше обозначить, что данные отсутствуют, чем случайно оставить ячейку пустой. Пустые ячейки — это всегда дополнительная работа для аналитика, ему придется догадываться, какое значение там должно стоять. А если данные отсортированы по какому-либо столбцу, это может быть очень сложно.

1. Рекомендуется хранить только одно значение в ячейке

Ячейки должны содержать информацию только об одном свойстве. Не рекомендуется записывать в нее множество значений.

Например, имеется колонка "позиция на карте", которая записывается "широта-долгота". Будет лучше разбить этот столбец на два: отдельно широта, отдельно долгота. Или если необходимо записать "500м", то лучше записать "500" в один столбец, а единицы измерения "м" вынести в отдельный.

Также, не рекомендуется объединять ячейки. Это может выглядеть красиво, но в итоге нарушается правило "не оставлять пустых ячеек"

1. Каноничный внешний вид

Лучшее расположение данных — это один большой прямоугольник со строками, отвечающими за объекты, и столбцами, отвечающими за переменные. Первая строка должна содержать названия переменных, но ни в коем случае не нужно использовать больше одной строки под эту цель.

Некоторые датасеты не будут хорошо выглядеть в одной прямоугольной таблице, тогда их рекомендуется представить в виде нескольких небольших таблиц. Лучше всего хранить каждую таблицу в отдельном файле; с таблицами разбросанными по файлу работать не очень удобно, а также трудно сделать экспорт в CSV файл. Если существует необходимость хранить разные таблицы на разных рабочих листах, то необходимо следить за них структурой.

1. Рекомендуется создавать словарь

Полезно иметь отдельный файл, который объясняет значения всех переменных. Оформляется он также в виде прямоугольной таблицы, так его сможет использовать аналитик в своих целях.

Такой файл может содержать:

* название переменной, как файле с данными
* версию названия переменной, которая используется в визуализации
* более длинное значение переменной с пояснениями
* единицы измерения
* минимальные и максимальные значения

Еще можно написать информацию о самих данных и создать ReadMe файл с описанием проекта.

## Алгоритмы нечеткого поиска

Нечеткий поиск — это очень полезная функция при работе с естественным языком. При этом, чтобы реализовать его эффективно, необходимо потратить гораздо больше сил по сравнению с поиском по точному совпадению.

Задачу нечеткого поиска можно представить так:

«По заданному слову найти в тексте или словаре размера **n** все слова, совпадающие с этим словом (или начинающиеся с этого слова) с учетом **k** возможных различий».

Главной характеристикой таких алгоритмов является метрика – функция расстояния между двумя словами, которая позволяет оценить сходство данных слов. Наиболее популярные метрики — это расстояния Хемминга, Левенштейна и Дамерау-Левенштейна. Однако расстояние Хемминга на практике является практически бесполезным.

### Расстояние Левенштейна

#### Общее описание

Имеется две строки S1 и S2. Необходимо превратить одну строку во вторую, используя три операции:

* I: Вставка символа в любую позицию;
* D: Удаление любого символа;
* R: Замена любого символа

Обозначим за d(S1, S2) минимальное количество вышеперечисленных операций для перевода одной строки в другую.

#### Алгоритм Вагнера-Фишера

Искомое расстояние находится через функцию D (M, N), которая находит редакционное расстояние между подстроками S1[0...M] и S2[0...N]. Таким образом, полной редакционное расстояние равно расстоянию подстрок полной длины: d (S1, S2) = DS1, S2(M, N).

Очевидно следующее:

* D (0,0) = 0 – пустые строки всегда совпадают;
* D (i, 0) = i, D (0, j) = j – из пустой строки можно получить любую, добавляя нужное количество символов.

Общий же случай выглядит следующим образом:

D (i, j) = D (i-1, j-1), если S1[i] = S2[j],

иначе D (i, j) = min (D (i-1, j), D (i, j-1), D (i-1, j-1)) + 1.

Здесь выбирается наиболее выгодное действие: удалить символ (D (i-1, j)), добавить (D (i, j-1)), или заменить (D (i-1, j-1)).

Для работы алгоритмы не требуется много памяти, достаточно хранить только два столбца: текущий (D (i, \*)) и предыдущий (D (i-1, \*)). Но если необходимо получить перечисление операций для перевода с их параметрами (редакционное предписание), понадобится матрица в полном объеме. Начиная из правого нижнего угла, матрицы (M, N) необходимо идти в левый верхний, на каждом шаге выбирая минимальное из трёх значений:

* если минимально (D (i-1, j) + 1), добавляем удаление символа S1[i] и идём в (i-1, j);
* если минимально (D (i, j-1) + 1), добавляем вставку символа S1[i] и идём в (i, j-1);
* если минимально (D (i-1, j-1) + m), где m = 1, если S1[i]! = S2[j], иначе m = 0; после чего идём в (i-1, j-1) и добавляем замену если m = 1.

Здесь (i, j) — клетка матрицы, в которой мы находимся на данном шаге. Если минимальны два из трёх значений (или равны все три), это означает, что есть 2 или 3 равноценных редакционных предписания. В итоге потребуется O(|S1|\*|S2|) времени и O(|S1|\*|S2|) памяти. [4]

### Алгоритм шинглов

Данный алгоритм состоит из 5 этапов:

1. канонизация текста;
2. разбиение на шинглы;
3. вычисление хэшей шинглов с помощью 84х статических функций;
4. случайная выборка 84х значений контрольных сумм;
5. сравнение, определение результата.

#### Канонизация текста

На данном этапе необходимо очистить текст от не нужного «мусора», а именно от предлогов, союзов, знаков препинания и прилагательных, так как они не несут смысловой нагрузки. Далее нужно привести все существительные к именительному падежу, единственному числу или же оставить только корень.

Конкретных правил для канонизации текста нет, здесь можно применять широкий спектр экспериментов. Главное на выходе иметь очищенный текст, готовый к сравнению.

#### Разбиение на шинглы

Шинглы это подпоследовательности, выделенные из текста. Для формирования подпоследовательностей есть следующие правила: длина одной равна десяти словам, идущим подряд, и каждый последующий шингл сдвигается на одно слово, то есть шингл с номером N имеет 9 общих слов с шинглами N – 1 и N + 1.

В результате получается массив шинглов, длиной равной количеству слов в тексте минус 10 (длине шингла) + 1.

#### Вычисление хэшей шинглов с помощью 84х статических функций

Суть алгоритма заключается в сравнении случайной выборки хешей шинглов двух текстов. От сюда вытекает и проблема алгоритма – количество сравнений. Чем больше шинглов – тем больше операций сравнения нужно выполнить, причем их количество растет экспоненциально.

Каждому шинглу ставится в соответствие 84 значения хеша, посчитанные через различные функции (SHA256, MD5, CRC32 и т.д.). В результате имеем двумерный массив из 84х строк, характеризующих соответствующий хеш.

#### Случайная выборка 84х значений контрольных сумм

Из полученных наборов случайным образом выбираются 84 значения для каждого текста и далее они сравниваются в соответствии функции хеша, через которую он был получен. Получается, нужно выполнить всего 84 операции сравнения.

Можно выбирать не случайным образом, а, например, самое минимальное значение для каждой строки. Тогда на выходе будет иметься набор из минимальных значений хешов шинглов для каждой функции.

#### Сравнение, определение результата

Это последний этап. Здесь сравниваются 84 элемента первого текст с таким же количеством элементов второго текста. Далее считается отношение одинаковых значений, после делается заключение. [5]

### Алгоритм N-грамм

Довольно старый и наиболее распространенный метод, из-за своей простой реализации и хорошей производительности. Главный принцип алгоритма – «Если слово А совпадает со словом Б с учетом нескольких ошибок, то с большой долей вероятности у них будет хотя бы одна общая подстрока длины N». Подстроки длины N – это и есть N-граммы.

Слово разбивается на N-граммы и попадает в списке для каждой N-граммы. Слова из другого текста также разбиваются на N-граммы. Затем происходит последовательный перебор списка слов, содержащих каждую подстроку.

На практике, самым часто используемым являются триграммы – подстроки длины 3. Выбор большего значения N ведет к ограничению на минимальную длину слова, при которой уже возможно обнаружение ошибок. [6]

## Алгоритмы идентификации таблиц

### Rule-based алгоритмы

#### TIRS

Алгоритм TIRS (Идентификация и реконструкция таблиц в электронных таблицах) рассматривает пространственное расположение таблиц на листе. Эмпирическими методами были определены типичные виды таблиц (рис. 1). Затем, используя наборы правил и эвристик, TIRS определяет эти таблицы на заданном листе

Важной идеей алгоритма является определение границы таблицы. На рисунке 1 они отмечены серым цветом. Границей обычно является заголовок или левая сторона. Главной функцией границ является помощь в определении начала новой таблицы, не важно, горизонтальной или вертикальной.

TIRS пытается определить таблицы даже не смотря на наличие помех. Как показано на рисунке 1, алгоритм берет во внимание возможное разбиение таблицы на части. Более того, TIRS замечает неправильные обозначения, которые встречаются в заголовках и теле таблицы.

Тем не менее, TIRS является достаточно плохим алгоритмом обнаружения таблиц. Одним из его главных недостатков является то, что он не использует структурное представление, например, в виде графа. Это следствие сложных правил, которые сложно отлаживать и расширять. [7]

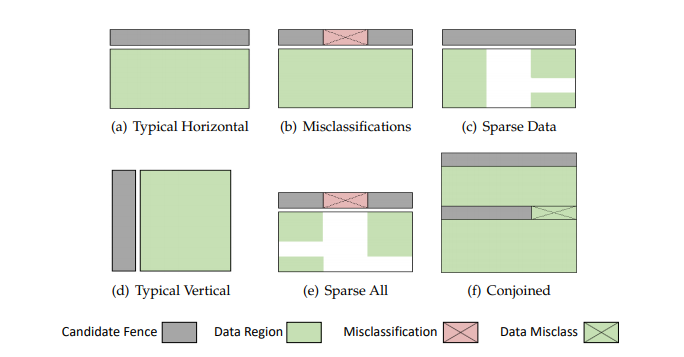


Рисунок 1

#### Удаляй и властвуй

Данный подход полагается на графовую модель. Преимуществом данного подхода над предыдущим является то, что его правила выходят за рамки непосредственных соседей области. Другими словами, он обрабатывает гораздо больше информации чем TIRS. Однако, алгоритм «Удаляй и властвуй» не использует вертикальные границы. Ему важны только заголовки и расположение данных. Тем не менее, алгоритм использует и упрощает множество правил, используемых в TIRS.

Удаляй и властвуй связывает листы с одной или многими таблицами, различными макетами и произвольными условностями. Однако, транспонированные таблицы вне зоны его работы, потому что встречаются они достаточно редко. То есть, алгоритм предполагает, что заголовки ячеек находятся сверху, а данные расположены ниже.

Следующее фундаментальное утверждение определяет данный алгоритм. Удаляются вершины, которые сохраняют вертикальную структуру внутри таблицы. Другими словами, приоритет верхних и нижних вершин больше, чем у левых и правых.

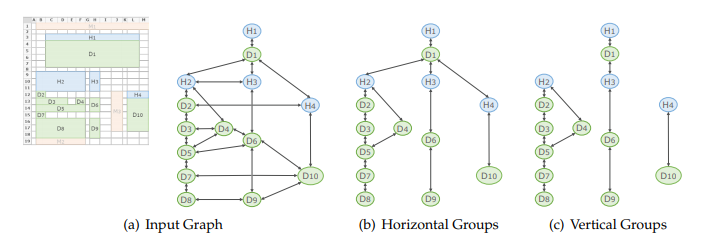


Рисунок 2

На рисунке 2 изображен лист с горизонтальным выравниванием таблиц, в графовом представлении получаются левые и правые вершины. Обычно эти вершины соединяют данные из разных таблиц, но также могут находиться внутри одной таблицы. Но для будущей обработки нужны еще верхние и нижние вершины. Удаление левых и правых вершин должно в основном влиять на внутретабличные соединения. Однако, для удаления межтабличных соединений нужно делать предварительные подсчеты, чтобы не удалить вершины, между которыми расстояние равно 0, т.е. между таблицами нет пустой колонки.

Вышеописанное действие по отбросу левых и правых вершин является первым шагом алгоритма «удаляй и властвуй». Получившиеся связанные компоненты относятся к так называемым «горизонтальным группам». Нужно обратить внимание, что не всегда будут получаться несколько таких групп. Например, на рисунке 2(b) есть всего один связанный компонент после проделанной операции.

Несмотря на это, на следующем шаге каждая горизонтальная группа будет разбиваться вертикально. Для этого будут использоваться заголовки в качестве разделителя. В конечном итоге, будут получены подгруппы, которые называются «вертикальные группы». Они уже должны напоминать валидные таблицы. Однако, на последних шагах алгоритм «удаляй и властвуй» пытается сделать еще несколько корректировок, которые могут улучшить качество распознавания таблиц.[8]

### Генетический алгоритм

Данный алгоритм выходит за рамки фиксированных правил. Это адаптивный и гибкий подход, который может подстраиваться под специфику ранее не встречаемого датасета в электронной таблице. Переход от алгоритма TIRS до генетического достаточно простой и требует минимальных усилий. Следовательно, был разработан почти автоматический алгоритм, который настраиваться на базовых примерах.

Здесь также используется графовая модель для представления данных на исходном листе. Далее, идет поиск по оптимальному разделению графа, основанному на целевой функции. С помощью этой функции оценивается качество потенциальных разделений. Зачем, используя генетический алгоритм, делается эффективных обход исследуемой области, с целью обнаружения наилучшего разбиения (глобальный оптимум). Логично предположить, что такое разбиение будет состоять из связных компонентов, которые в точности соответствуют таблицам на листе.

Целевая функция использует 10 метрик:

1. Коэффициент заголовков;
2. Коэффициент данных;
3. Являются ли данные разделенными;
4. Являются ли заголовки разделенными;
5. Находятся ли однородные данные в одной колонке;
6. Количество других валидных заголовков;
7. Оценка данных над заголовками;
8. Среднее количество соседних пустых колонок;
9. Среднее количество соседних пустых строк;
10. Оценка пересечений.

Сама функция имеет следующий вид:

obj(P, M, w) := w10 ∗ M10(*P*) + ∑Vi∈P fit(Vi , M, w), здесь P это множество всех найденных разбиений Vi, M – множество всех метрик Mi, wi – вес каждой метрики.

Функция собирает вместе несколько метрик, которые отвечают за различные аспекты. Потенциальное разбиение рассматривается как единое целое, также как и отдельные части, из которых оно состоит. Тем не менее, некоторые параметры важнее других. Следовательно, используемые метрики имеют свои веса, основанные на их актуальности. Вычисляются же веса автоматически. Для этого используются тренировочные наборы электронных таблиц. После этого можно применять целевую функцию к нужному датасету. [9]

# Список литературы

1. <https://excelwithbusiness.com/blog/microsoft-excel-versus-apples-numbers-who-prevails/>
2. https://excel2.ru/articles/sovety-po-postroeniyu-tablic-v-ms-excel
3. Data Organization in Spreadsheets Karl W. Bromana and Kara H. Woob (журнал)
4. <https://habr.com/ru/post/117063/>
5. <https://habr.com/ru/post/65944/>
6. <https://habr.com/ru/post/114997/>
7. Layout Inference and Table Detection in Spreadsheet Documents (диссертация) 108-109с.
8. Layout Inference and Table Detection in Spreadsheet Documents 109-110c
9. Layout Inference and Table Detection in Spreadsheet Documents 114c