

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Базовая кафедра «Интеллектуальные системы управления»

УТВЕРЖДАЮ
Заведующий кафедрой ИСУ
Ю. Ю. Якунин
подпись инициалы, фамилия
«_____» 2020 г.

БАКАЛАВРСКАЯ РАБОТА

27.03.03 Системный анализ и управление

Библиотека для исправления опечаток с учетом контекста

Руководитель: доцент каф. ИСУ, канд.техн.наук А.А. Даничев
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник: _____
подпись, дата

А.О. Бондаренко
ициалы, фамилия

Красноярск 2020

Оглавление

ВВЕДЕНИЕ.....	4
Глава 1. Способы исправления опечаток.....	6
1.1 Опечатки	6
1.2 Методы исправления опечаток	7
1.4 Расстояние Дамерау-Левенштейна	8
1.5 Алгоритм Бойера-Мура	9
1.6 Метод N-грамм.....	11
Выводы по главе 1.....	12
Глава 2	13
2.1 Хранение словаря и языковой модели.	13
2.2 Метод нахождения слова для исправления опечатки	13
2.3 Выбор языковой модели	14
2.4 Выбор языка для написания библиотеки.....	14
Выводы по второй главе	14
Глава 3	16
3.1 Создание словаря и языковой модели.....	16
2.2 Метод нахождения кандидатов для исправления опечатки	18
2.3 Выбор кандидата для замены слова с опечаткой с учетом контекста.....	20
2.5 Создание библиотеки.....	21
2.6 Создание текстового редактора для демонстрации работы библиотеки...	22
Выводы по 3 главе.....	24
ЗАКЛЮЧЕНИЕ	26
БИБЛИОГРАФИЧЕСКИЙ СПИСОК	27

РЕФЕРАТ

Бакалаврская работа по теме «Библиотека для исправления опечаток с учетом контекста» содержит 27 страниц текстового документа, 12 рисунков, 7 использованных источника.

КЛЮЧЕВЫЕ СЛОВА: ИСПРАВЛЕНИЕ ОПЕЧАТОК, ИСПРАВЛЕНИЕ ОПЕЧАТОК С УЧЕТОМ КОНТЕКСТА.

Объектом исследования в данной работе модель исправления опечаток с учетом контекста, предметами – алгоритмы исправления опечаток, языковая модель.

Целью работы заключается в изучении алгоритмов исправления опечаток и написании Библиотека для исправления опечаток с учетом контекста.

Чтобы достичь поставленной цели работы были поставлены следующие задачи:

- Провести анализ алгоритмов исправления опечаток;
- Выбор алгоритма исправления опечаток и его доработка;
- Написание модели языка;
- Создание библиотеки.

ВВЕДЕНИЕ

Опечатка — ошибка в тексте, обычно в результате невнимательности наборщика. Чаще всего, в результате опечатки нарушается порядок букв в слове (*быт* вместо *быть*), одна буква исчезает из слова (*чловек* вместо *человек*) или одна буква заменяется другой (*чтатья* вместо *статья*).

Со времен создания первых машин для печатания текста люди допускают опечатки в тексте. Изначально опечатки исправлялись вручную в каждом экземпляре отпечатанного текста. По мере развития типографского дела по инициативе издателя Габриэля Пьерри стал применяться более простой метод внесения правок, заключающийся в перечислении опечаток в конце книги. В быстро развивающемся мире, где количество печатного текста неминуемо росло, росло и количество допускаемых опечаток.

В 1960-х годах в инженерной среде возникла потребность возникла потребность в автоматическом исправлении опечаток. Коррекция опечаток — одна из проблем в области обработки ЭВМ естественного языка. Универсального решения данной проблемы до сих пор не существует, однако на протяжении всей истории коррекция орфографии отражала актуальные задачи прикладной лингвистики и вбирала в себя самые новые вычислительные методы. Развитие этой дисциплины определяется двумя основными технологическими потребностями: удобством дальнейшей обработки текста (оптимизация поисковой выдачи) и удобством набора для пользователя (проверка и коррекция орфографии в текстовых редакторах). Так в 1980-х годах появляются первые коммерческие продукты для редакции текста включающие в себя модули для поиска и исправления опечаток.

В современном мире с распространением персональных компьютеров продукты по исправлению опечаток становятся все более популярными.

Большинство редакторов текста включают в себя модули исправления опечаток, а поисковые системы в веб-интерфейсах используют алгоритмы автоматического исправления опечаток с учетом контекста. Например, такие алгоритмы используются для функций наподобие «Возможно вы имели в виду ...» в тех же поисковых системах.

Глава 1. Способы исправления опечаток

1.1 Опечатки

Чаще всего опечатки возникают из-за невнимательности, причиной которой может служить усталость, отсутствие интереса к работе, отвлекающие факторы или высокая скорость работы. При наборе текста в большинстве случаев опечатки возникают из-за случайности. По статистике пользователи совершают опечатки приблизительно в 10-15% случаях. При этом 83,6% запросов имеют одну ошибку, 11,7% – две, 4,8% – более трёх. Контекст важен в 26% случаев. Орфографическая ошибка сама по себе не считается опечаткой, однако может являться результатом опечатки. Случается, что опечатка влечёт замену одного слова другим, в результате чего текст либо полностью утрачивает смысл, либо приобретает новый. Нередко это создает комический эффект. Опечатки можно разделить на следующие виды:

- Нарушение порядка букв в слове;
- Пропуск букв в слове;
- Добавление в слово лишней буквы;
- Замена одной буквы другой;
- Ошибка при написании заглавных или строчных букв;
- Неправильная расстановка дефисов и тире;
- Исаженная раскладка.

В большинстве случаев опечатки в документах неприемлемы, что может привести к тому, что документ будет считаться недействительным и

необходимо будет создавать новый документ, что может повлиять на сроки начала вступления в силу документа.

Для исправления опечаток в большинстве программ для работы с текстом встроены полуавтоматические корректоры, указывающие на опечатку и предлагающие набор слов для замены. В некоторых программах и сервисах для исправления текста используется автокорректор автоматически заменяющий слово с опечаткой, но для его корректной работы требуется высокая точность, большой словарь и работа с контекстом.

1.2 Методы исправления опечаток

В общем виде механизм контекстно независимого исправления опечаток основывается на модели ошибок и словаре, содержащем список слов без опечаток. В качестве модели ошибок обычно используют методы N-грамм, модель Бриля-Мура либо редакционное расстояние. Модели ошибок используют алгоритмы нечеткого поиска. Они являются основной системой проверки орфографии в текстовых редакторах и поисковых систем.

Для работы контекстно независимого механизма также требуется языковая модель, представляющая вероятностное распределение на множестве словарных последовательностей. Возможность контекстно зависимого исправления позволяет более точно предсказывать слово для замены опечатки, но усложняет построение корректора из-за использования языковой модели, которая может потреблять больше ресурсов чем корректор без использования контекста. Если обратить внимание на статистике, то 74% всех опечаток можно исправить без учета контекста, что обуславливает существенную пользу от контекстно независимого автокорректора.

Реализация нечеткого поиска сложнее поиска по точному совпадению, но все компенсирует его крайняя эффективность. Задачу алгоритма нечеткого

поиска можно поставить как поиск в тексте или словаре объема на все слова, совпадающие с этим словом или включающие это слово. Алгоритмы нечеткого поиска характеризуют метрикой – функцией расстояния между двумя словами, позволяющей оценить степень их сходства.

1.4 Расстояние Дамерау-Левенштейна

Расстояние Левенштейна — это метрика, измеряющая разность между двумя последовательностями символов. Рассчитывается как минимальное количество односимвольных операций необходимых для превращения одной последовательности символов в другую. Часто применяется для исправления ошибок в слове и сравнения текстов. Для корректного расчета ошибок операциям вставки, замены, удаления и пропуска символа устанавливают различные цены (w), отображающие разную вероятность ошибки при вводе текста.

- $w(a, b)$ — цена замены символа a на символ b
- $w(\varepsilon, b)$ — цена вставки символа b
- $w(a, \varepsilon)$ — цена удаления символа a

Пусть s_1 и s_2 — две строки (длиной M и N соответственно) над некоторым алфавитом, тогда редакционное расстояние (расстояние Левенштейна) $d(s_1, s_2)$ можно подсчитать по следующей рекуррентной формуле $d(s_1, s_2) = D(M, N)$, где

$$D(i,j) = \begin{cases} 0, & i = 0, j = 0 \\ i, & j = 0, i > 0 \\ j, & i = 0, j > 0 \\ \min \{ & \\ D(i, j - 1) + 1, & \\ D(i - 1, j) + 1, & j > 0, I > 0 \\ D(i - 1, j - 1) + m(s_1[i], s_2[j]) & \\ \} & \end{cases} \quad (1)$$

Где $m(a, b)$ равна нулю, если $a = b$ и единице в противном случае, $\min\{a, b, c\}$ возвращает наименьший из аргументов.

Шаг по i символизирует удаление из первой строки, по j – вставку в первую строку, шаг по обоим индексам символизирует замену символа или отсутствие изменений (1).

Если к списку разрешённых операций добавить транспозицию (два соседних символа меняются местами), получается расстояние Дамерау — Левенштейна. Для неё также существует алгоритм, требующий $O(MN)$ операций. Дамерау показал, что 80 % ошибок при наборе текста человеком являются транспозициями.

1.5 Алгоритм Бойера-Мура

Алгоритм Бойера-Мура представляет собой алгоритм общего назначения для поиска подстроки в строке. Преимуществом алгоритма являются предварительные вычисления над шаблоном строки, что позволяет сравнивать шаблон с исходным текстом не во всех позициях - часть проверок пропускаются как заведомо не дающие результата.

Алгоритм основан на идеи сканирование слева направо. Совмещается начало строки и шаблона, проверка начинается с последнего символа шаблона.

При совпадении символов производится проверка предпоследнего символа и т.д. Если все символы совпали значит подстрока найдена. В случае несовпадения любого символа происходит сдвиг до первого совпадения стоп-символа с шаблоном, часто стоп символом является последний символ шаблона. На рисунке 1 представлено графическое пояснение к данной части алгоритма.

Строка:	* * * * * * * * * л * * * * *
Шаблон:	ко ло кол
Следующий шаг:	ко ло кол

Рисунок 1- Смещение шаблона до стоп-слова

Далее происходит создание суффикса- сравнение символа, идущего перед стоп-символом и предпоследним символом шаблона и т.д. В случае несовпадения суффикса и части шаблона происходит сдвиг в право до совпадения суффикса с частью шаблона. Если последующие части шаблона не совпадают с суффиксом, то из суффикса исключается левый символ, а не совпавший символ К запоминается. Происходит сдвиг в право до первого совпадения суффикса с частью шаблона в которой слева нет символа К. На рисунке 2 представлено графическое пояснение к данной части алгоритма.

Строка:	* * * * * е ка * * * * *
Шаблон:	ск ла ка лка
Следующий шаг:	ск а лка лка

Рисунок 2- смещение шаблона до суффикса

В данном случае произошел сдвиг до совпадения суффикса «ка» перед которым нет символа «л». Суффикс «ека» не совпадает с шаблоном, а в левой части шаблона не осталось символов, содержащих совпадения суффикса. Далее происходит сдвиг в право до первого стоп-символа и алгоритм создания суффикса повторяется.

1.6 Метод N-грамм

Метод N-грамм часто используется, так как его реализация крайне проста, и он обеспечивает хорошую производительность. Метод основан на совпадении подстрок длины N если слово А совпадает со словом В с учетом нескольких ошибок. При поиске слово с опечаткой разбивается на N-граммы, затем производится последовательный перебор списка слов на содержание таких подстрок. На рисунке 3 демонстрируется разбиение слова на триграммы. Слово содержащее наибольшее количество N-грамм вероятнее всего является правильным словом.

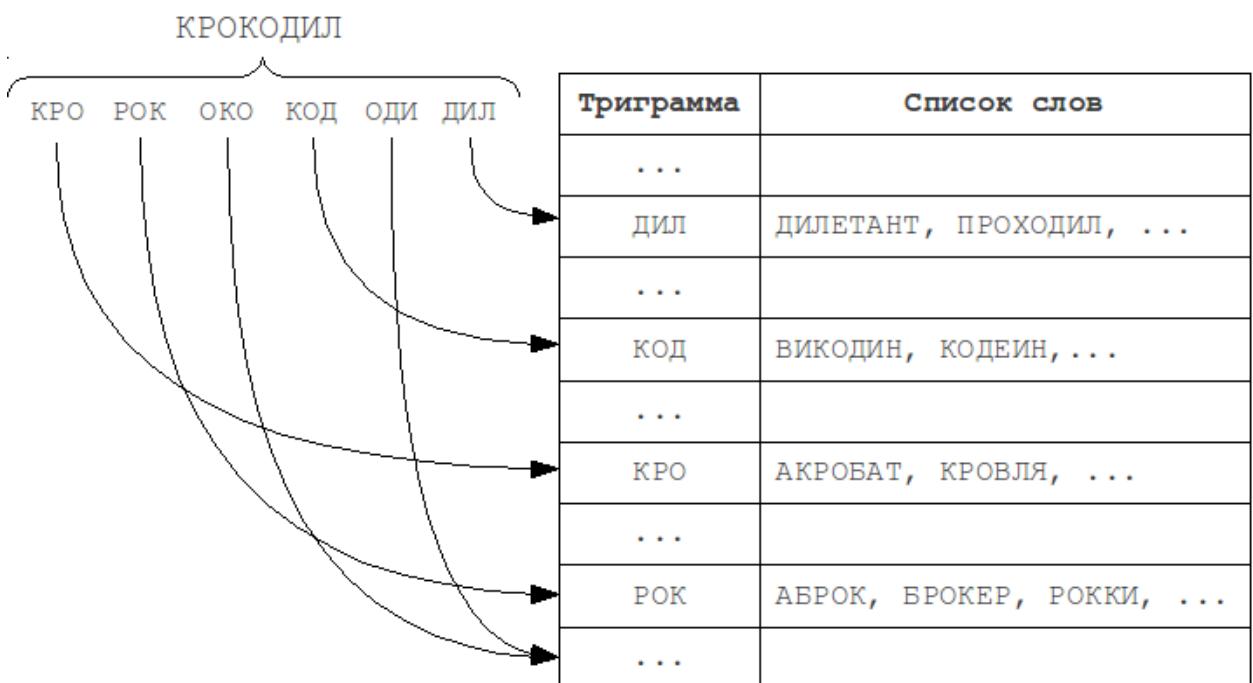


Рисунок 3 – пример работы алгоритма триграмм

Чаще всего используют на практике триграммы – подстроки длиной в три символа. Выбор большего значения N приводит к ограничению длины слова, в котором возможно обнаружить опечатку. Стандартные алгоритмы N-грамм находят не все возможные слова с ошибками. Если при использовании

триграмм в слове из пяти букв допустить опечатку в третьей букве, то не получится ни одной триграммы совпадающей с искомым словом.

Выводы по главе 1

Исследуется проблема возникновения опечаток и возможности их дальнейшего исправления. Проблему возникновения опечаток решить не представляется возможным так как опечатки совершают человек, и эта проблема существовала еще на заре книгопечатания. Исправление опечаток может осуществляться как различными методами нечеткого поиска, так и линейным поиском. Для исправления опечаток был выбран метод нечеткого поиска с помощью N-грамм. Такое решение было принято в виду того, что данный метод прост в исполнении и больших возможностей для усовершенствования. Важным недостатком данного метода является не способность найти правильной слово в словаре. Данний недостаток возникает в уникальных случаях и для его устранения можно использовать линейный поиск так как такие случаи возникают очень редко.

Глава 2

2.1 Хранение словаря и языковой модели.

Для работы библиотеки для исправления опечаток с учетом контекста необходимо создать словарь и языковую модель. Для упорядочивания и хранения данных была выбрана система управления базами данных MySQL, позволяющая быстро считывать большие объемы данных, не потребляя большое количество вычислительной мощности компьютера. Также преимуществом MySQL является возможность удаленного подключения к базе данных и возможность одновременного подключения к базе данных неограниченного числа пользователей, что позволяет не хранить словарь и языковую модель на отдельном сервере. Как следствие сократится потребление вычислительной мощности компьютера и уменьшится объем памяти на запоминающем устройстве со стороны пользователя, так как словарь и языковая модель занимают значительный объем памяти.

2.2 Метод нахождения слова для исправления опечатки

Для исправления опечатки необходимо найти слово, в котором допущена опечатка, для этого необходимо проверить каждое слово посимвольно на наличие в словаре. Если при проверке слово отсутствует в словаре вероятно в нем допущена опечатка. Когда найдено слово с опечаткой необходимо найти из словаря кандидатов для исправления слова с опечаткой.

Для этого был выбран метод N-грамм, а в частности метод триграмм. Модификация работы данного метода имеет ряд преимуществ перед другими методами: простота и лучшая скорость поиска по словарю.

2.3 Выбор языковой модели

Модель языка отвечает за вероятность написания слов в определенном порядке. На сегодняшний день в основном используются два подхода: модели на основе N-грамм, а также модели на основе нейросетей. Для создания языковой модели был выбран метод триграмм. Предложение разбивается на части на месте знаков пунктуации, на полученных участках рассчитывается вероятность парного написания слов.

Данная модель способна выбрать из кандидатов на замену слова с опечаткой кандидата который чаще всего встречается в контексте соседних слов. Основным минусом данной модели является необходимость хранения матрицы размером M на M , где M - количество слов в словаре.

2.4 Выбор языка для написания библиотеки.

Для создания библиотеки использовался язык программирования C# в среде разработки Microsoft Visual Studio. C# является распространенным языком для написания библиотек, которые могут использоваться во многих языках программирования, что повлияло на выбор этого языка для написания библиотеки.

Выводы по второй главе

Выбор способа хранения словаря и языковой модели на отдельном сервере под управлением СУБД MySQL позволяет уменьшить объем памяти на

запоминающем устройстве пользователя. Также пользователь уменьшить нагрузку на вычислительные мощности необходимые для обработки запросов, обращенных к базе данных, так как вычисления происходят на удаленном сервере. Такое решение также позволяет использовать выбранную языковую модель. Использование языка программирования C# для написания библиотеки является очевидным так как библиотеки, написанные на этом языке, поддерживаются большим количеством языков программирования.

Глава 3

3.1 Создание словаря и языковой модели.

Общий вид словаря и языковой модели

Словарь представляет собой таблицу, состоящую из двух столбов в первый столбец, записывается уникальный номер, во второй столбец – уникальное слово, которое записано в новой строке. На рисунке 4 представлен вид хранения словаря.

num	word
1	я
2	не
3	что
4	в
5	и
6	ты
7	это
8	на
9	с
10	он
11	вы
12	да
13	как
14	мы
15	мне
16	а
17	меня
18	у
19	нет
20	так
21	но
22	то
23	все
24	его
25	тебя
26	за

Рисунок 4 – Словарь

Для исправления опечаток с использованием контекста также нужно хранить языковую модель, которая является матрицей, содержащей количество парных комбинаций слов необходимых для корректной работы библиотеки. На

рисунке 5 демонстрируется одна из таблиц в которой хранится языковая модель. Эти данные хранятся в таблицах размером (N - количество слов в словаре) строк на ($M=1001$) столбцов, где в первый столбец записаны все слова словаря, а остальные столбцы названы уникальными словами из словаря. Ограничение в 1001 столбец обуславливается структурой построения таблиц в MySQL не позволяющее создать больше 1017 столбцов в одной таблице.

word	я	не	что	в	и	ты	это	на	с	он
во	NULL									
возможно	NULL									
возьми	NULL									
бот	NULL									
времени	NULL									
время	NULL									
все	NULL									
всегда	NULL									
всего	NULL									
всёё	NULL									
всем	NULL									
всех	NULL									
всю	NULL									
вы	NULL									
где	NULL									
говорил	NULL									
говорит	NULL									
говорить	NULL									
говорю	NULL									
господи	NULL									
да	NULL									

Рисунок 5 – Языковая модель

Заполнение словаря и языковой модели

Для создания и заполнения словаря была написана программачитывающая данные из текстового документа и заполняющая таблицу. Так же написанная программа создает таблицы для хранения количества попарных комбинаций слов и заполняет эти таблицы. Изначально все ячейки кроме

первого столбца имеют значение единицы. Это позволяет избежать ситуации, когда какая-то пара слов не встречалась в обучающем тексте, из-за чего в таблице ячейка комбинаций таких слов будет равна нулю. Что не позволяет вероятному слову для исправления опечатки быть выбранным для исправления опечатки. Для заполнения таблиц количества попарных комбинаций слов происходит чтение текста, во временные переменные (first, second) записываются первое и следующее слово, в столбце word таблицы расчетов находится значение переменной first и в пересечении столбца со значением переменной second в соответствующей таблице прибавляется единица. После в переменную first переписывается слово из переменной second, а переменная second принимает значение следующего слова. При достижении знаков препинания переменные обнуляются и процесс чтения продолжается после знака.

2.2 Метод нахождения кандидатов для исправления опечатки

Когда найдено слово с опечаткой происходит разбиение слова этого слова на триграммы и выполняется поиск слов из словаря содержащих триграммы. Поиск происходит только для слов, в которых разница символов не более 3. Слова в словаре с наибольшим содержанием триграмм становятся кандидатами для замены слова с опечаткой.

При использовании метода поиска триграмм находящихся в слове, состоящего из пяти букв и имеющего опечатку в третьей букве, на рисунке 6 показано, что найти нужного кандидата не представляется возможным, так как при любом поиске из словаря не найдется необходимого слова, включающего хотя бы одну из триграмм находящихся в слове с опечаткой.

число
чи~~м~~ло
чи~~м~~
и~~м~~л
~~м~~ло

Рисунок 6 – Невозможность нахождения правильного кандидата

Во время использования метода поиска триграмм находящихся в слове, невозможно найти кандидата для замены слова с опечаткой состоящего из четырех символов в котором опечатка находится во втором или третьем символе данная проблема отражена на рисунке 7.

тире
т~~м~~ре
т~~м~~р
~~и~~ре

Рисунок 7- Невозможность нахождения правильного кандидата

Также во время использования метода поиска триграмм находящихся в слове, не предоставит кандидатов если опечатка добавления лишней буквы произойдет в словах, содержащих три буквы, что отражено на рисунке 8.

дом
до~~о~~м
до~~о~~
~~о~~м

Рисунок 8 - Невозможность нахождения правильного кандидата

Для нахождения кандидатов для всех вышеперечисленных случаев используется алгоритм удаления одной буквы. Алгоритм начинает работу после поиска кандидатов используя поиск слов, включающих триграммы, в словаре. Алгоритм удаления работает по принципу удаления одного символа, за

исключением первого и последнего символа, из слова с опечаткой. После чего происходит поиск из словаря по новым триграммам среди слов, содержащих меньше символов, чем слово с опечаткой. Если в слове из словаря количество символов равно количеству символов в слове с опечаткой, то из него удаляется символ по счету равный удаленному из слова с опечаткой. Затем слова из словаря с удаленным символом также проверяются на содержание новых триграмм основные этапы данного алгоритма представлены на рисунке 9. При нахождении слов в словаре включающих новые триграммы с учетом удаления слово из словаря становится кандидатом для замены слова с опечаткой.

ЧИСЛО

ЧИМЛО

ЧИЛО

ЧИЛ

ИЛО

Рисунок 9 – Алгоритм удаления

2.3 Выбор кандидата для замены слова с опечаткой с учетом контекста.

Чтобы правильно заменить слово с опечаткой необходимо выбрать наиболее подходящего из кандидатов. Если перед словом находится другое слово данного предложения оно записывается во временную переменную, тоже происходит при наличии следующего слова в предложении. Использование при расчетах как предыдущего, так и последующего слова обуславливается сложностью Типологии порядка слов русского языка, которая допускает разный порядок написания одних и тех же слов в предложении. Например, «Летят надо мной журавли» и «Журавли летят надо мной».

После нахождения слов, окружающих слово с опечаткой, используется таблица расчетов вероятностей, в которой находится информация о наиболее часто используемых комбинациях слов. Поиск в таблице происходит по столбцу которой характеризует связные слова для данного кандидата. Найдется кандидат, с большим количеством комбинаций с предыдущим и следующим словами, затем происходит замена слова с опечаткой на такого кандидата. В случае, когда опечатка допущена в первом или последнем слове участка предложения ограниченного знаками препинания выбор кандидата происходит выбор кандидата. В таком случае на выбор кандидата будет влиять только одно значение количества комбинаций из таблицы, которое характеризует частоту написания исправленного слова с опечаткой и соседнего слова.

2.5 Создание библиотеки

Библиотека включает в себя две функции: функцию поиска слов с опечаткой и функцию исправления опечатки.

Функция поиска (`find`) опечатки принимает на вход одно слово. После происходит поиск этого слова в словаре, если слово отсутствует в словаре вероятнее всего в нем допущена опечатка. Функция возвращает бинарное значение, в котором записан результат поиска. `True` если слово присутствует в словаре, `false` когда совпадений в словаре нет.

Для исправления опечатки используется функция (`correction`) исправления опечатки. Во время работы данной функции библиотека принимает на вход три слова из предложения записанные в переменные типа `string` (`first`, `second`, `third`). Переменная `second` содержит слово с опечаткой, переменная `first` содержит предыдущее слово, `third` следующее. Принимаемые

слова не должны разделяться знаками препинания. Возможны случаи, когда библиотека примет одно или два слова, это обуславливается малым участком предложения, ограниченного знаками препинания. В случае с двумя словами происходит выбор кандидата с учетом контекста. Если библиотека принимает одно слово, то исправление опечатки происходит путем выбора кандидата без учета контекста, который посимвольно наиболее схож со словом, содержащим опечатку. Функция возвращает массив слов, которые подходят для замены слова с опечаткой.

2.6 Создание текстового редактора для демонстрации работы библиотеки

Для демонстрации работы библиотеки был написан простейший текстовый редактор, к которому подключена библиотека. На рисунке 10 показан текстовый редактор.



Рисунок 10 – Текстовый редактор

При нажатии на кнопку «найти опечатки» происходит пословное считывание текста. Каждое слово проходит проверку на наличие в словаре. Если слово отсутствует в словаре, то оно подчеркивается и считается опечаткой. На рисунке 11 изображен результат поиска опечаток.

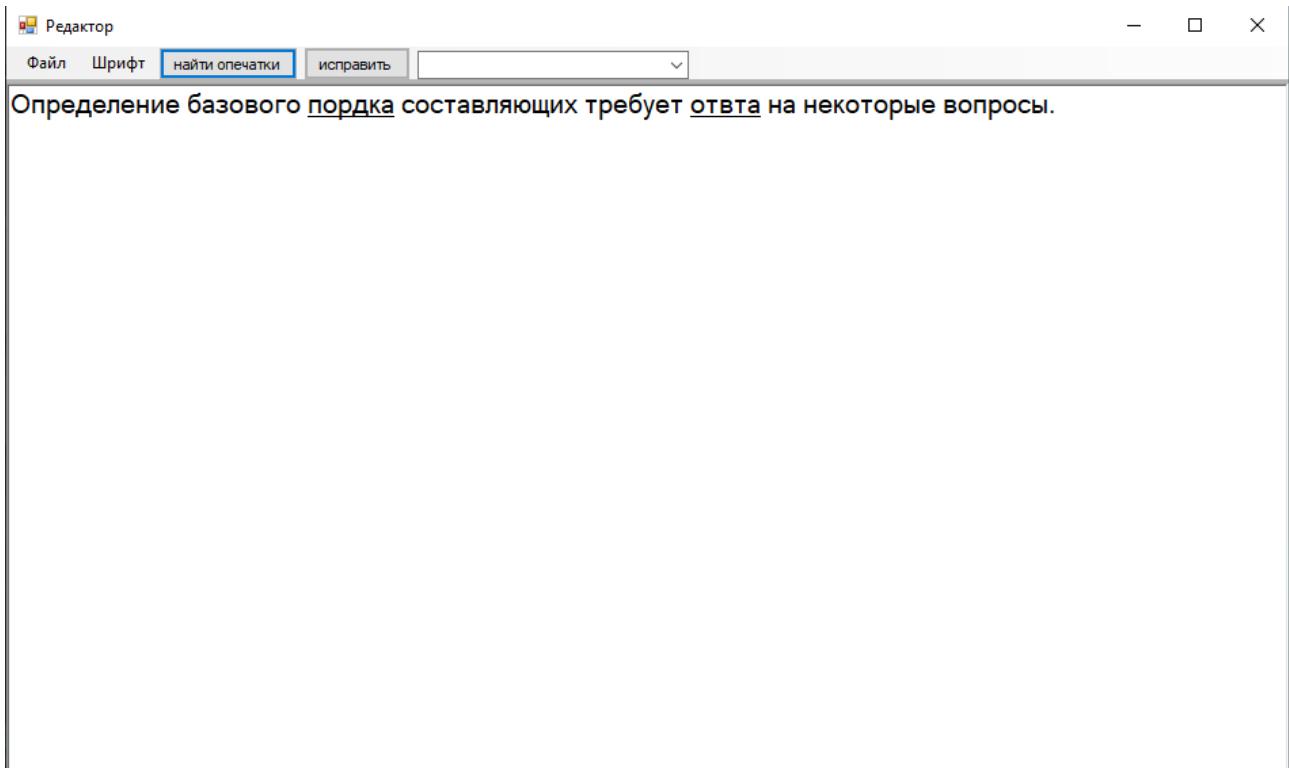


Рисунок 11 – Поиск опечаток

Найденные слова с опечатками, а также предыдущее и следующее слова если такие имеются, передаются в библиотеку. Библиотека возвращает список слов, наиболее подходящих для замены. Для выбора слова которое заменит опечатку используется выпадающий список, в котором необходимо выбрать слово для замены. На рисунке 12 изображен список кандидатов, подходящих для замены. Опечатки исправляются по очереди начиная с начала текста.

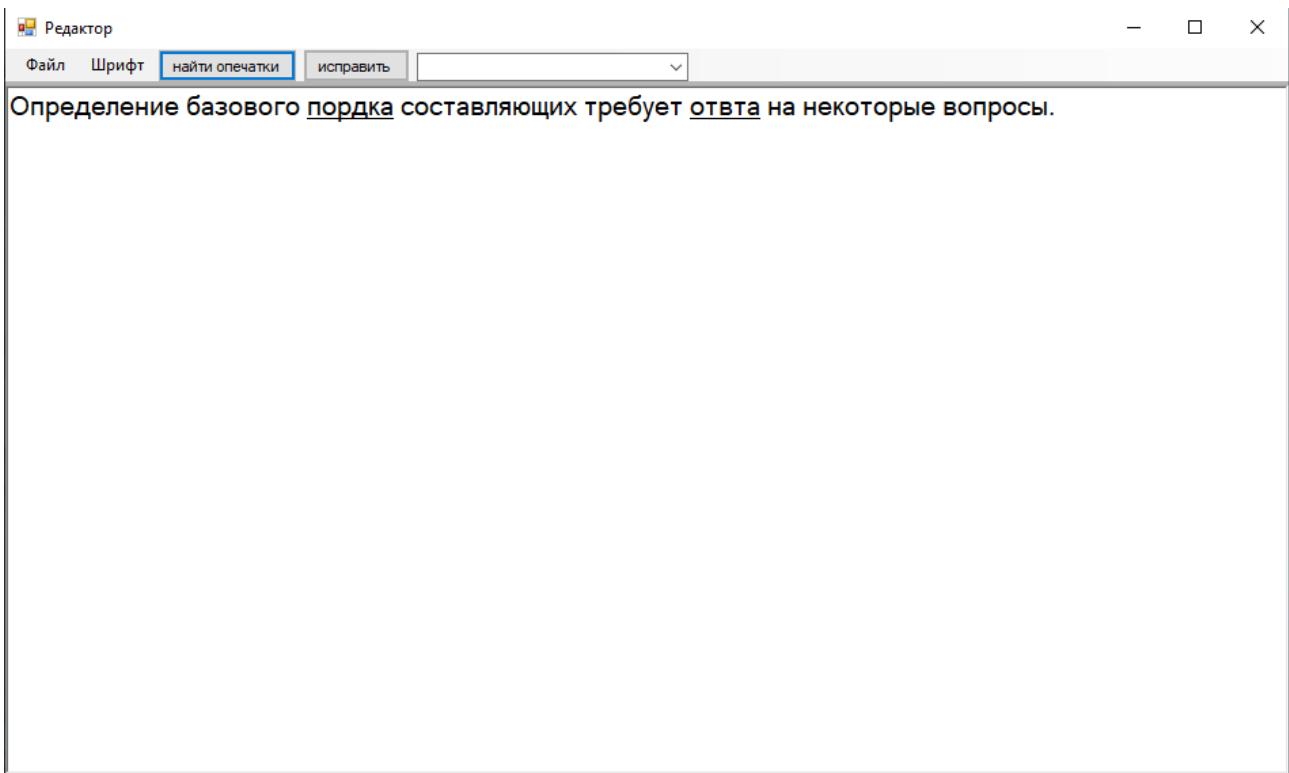


Рисунок 12 – Список кандидатов для замены

После выбора кандидата из списка происходит замена слова содержащего опечатку. Затем происходит исправление следующего слова с опечаткой если такой имеется.

Выводы по 3 главе

В главе описывались процессы создания словаря и языковой модели, написания библиотеки и текстового редактора, демонстрирующего работу библиотеки. Для заполнения словаря и языковой модели также была написана программа, работающая с базой данных. Основной трудностью при работе с MySQL было большое время создания и заполнения таблиц количества парных комбинаций. Словарь содержит 360000 слов, этого количества достаточно для корректной работы библиотеки. Языковая модель представляет собой матрицу размером 360000 на 360000, которая разбита на 360 таблиц.

Написанная библиотека способна определить слово с опечаткой и предложить кандидатов для замены с учетом контекста. Написанный текстовый редактор позволяет оценить скорость работы библиотеки. Код библиотеки закреплен в приложении.

ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе были рассмотрены алгоритмы для исправления опечаток тексте с учетом и без учета контекста. Описаны методы создания языковой модели. Была создана программа для создания языковой модели, библиотека для исправления опечаток с учетом контекста и простейший текстовый редактор для демонстрации работы библиотеки, а также показан принцип работы выше перечисленных программ и библиотеки.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Шаврина Т.О. Методы обнаружения и исправления опечаток: исторический обзор. 2017.
2. Панина М.Ф., Байтин А.В., Галинская И.Е. Автоматическое исправление опечаток в поисковых запросах без учета контекста. Яндекс, Москва, Россия.
3. Ukkonen E. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science* 92 (1992), 191-211.
4. Седова А.Г. Тематическое моделирование русскоязычных текстов с опорой на леммы и лексические конструкции
5. Маслинский К.А. N-граммы. моделирование локального контекста Компьютерные методы анализа текста. НИУ ВШЭ Санкт-Петербург 2014.
6. Бойцов Л.М. Классификация и экспериментальное исследование современных алгоритмов нечеткого словарного поиска // Труды 6-ой Всероссийской научной конференции “Электронные библиотеки: перспективные методы и технологии, электронные коллекции” - RCDL2004, Пущино, Россия, 2004.
7. Myers E.W. A Fast Bit-Vector Algorithm for Approximate String Matching Based on Dynamic Programming, In *Journal of the ACM (JACM)*, volume 46(3), 1998, p. 395 – 415.

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Базовая кафедра «Интеллектуальные системы управления»

УТВЕРЖДАЮ
Заведующий кафедрой
ИСУ
Ю. Ю. Якунин

БАКАЛАВРСКАЯ РАБОТА

27.03.03 Системный анализ и управление

Библиотека для исправления опечаток с учетом контекста

Руководитель: А.А. Даничев
подпись, дата 25.06.10 доцент каф. ИСУ, канд.техн.наук
должность, ученая степень
инициалы, фамилия

Выпускник: ~~Бондаренко~~ 25.06.2020
подпись, дата

Красноярск 2020