Автономная некоммерческая общеобразовательная организация "Физтех-лицей"
(АНОО «Физтех-лицей» им. П.Л. Капицы)

XX научно-практическая конференция

«Старт в инновации»

«Автоматическая корректировка орфографии»

Выполнено:

Балдина Анастасия, 9 «И» класс

Кудрявцева Василиса, 9 «И» класс

Руководитель:

Мерзляков Алексей Владимирович

Московская область, г. Долгопрудный

2021 г.

СОДЕРЖАНИЕ

Введение

Область исследования: компьютерная лингвистика;

Цель: создать автоматический предиктивный корректировщик орфографии с применением собственных модификаций стандартных алгоритмов корректировки.

Задачи:

1. Изучить известные методы компьютерной лингвистики автоматической корректировки правописания с помощью Интернет-ресурсов;
2. Изучить работу приложения T9;
3. Практическая реализация алгоритма автоматического корректировщика орфографии с разработкой собственных модификаций;
4. Разработка пользовательского интерфейса для демонстрации работы алгоритма.

Методы:

1. Изучение литературы и Интернет-ресурсов;
2. Эвристический метод подбора параметров алгоритма, позволяющий улучшить качество его работы.

Актуальность данной работы обоснована тем, что все больше людей для общения используют мессенджеры и социальные сети для общения. Зачастую, при быстром наборе текста, мы допускаем ошибки и опечатки. Чтобы сделать набор текста более эффективным и комфортным, существуют предиктивные (предугадывающие) системы набора текстов, такие как T9 или iTap.

В ходе работы будет собрана и обобщена информация о предиктивных системах набора текстов и разработан концепт приложения. Приложение будет являться виртуальной клавиатурой, находящей ошибки в набранном слове и предлагающей исправления для него.

Литературный обзор

О самой работе известнейших систем T9 и iTap достаточно мало информации, однако можно разобраться в их истории и методах облегчение ввода:

T9 — предиктивная (предугадывающая) система набора текстов для мобильных телефонов. Название T9 происходит от англ. ***T****ext on* ***9*** *keys*, то есть набор текста на 9 кнопках.[1] Раньше, когда телефоны были кнопочными, ввод текста на телефонах происходил с помощью девяти кнопок и каждой букве или знаку соответствовало определенное количество нажатий на кнопку. Чтобы написать одно слово уходило достаточно много времени из-за необходимости соблюдать паузы между нажатием на кнопки. Это исправила система T9. Ее преимущество в том, что за счет готовой базы слов давались подсказки, и набирать можно было буквально цифрами. При наборе текста система T9 пытается предугадать, какое слово вы пытаетесь набрать, используя встроенный словарь ,но не заменяет их. T9 подставляет только те слова, которые содержат столько букв, сколько набрано на данный момент.

iTap имеет существенное отличие от системы T9 - пытается предугадать и более длинные слова, анализируя не только набранные буквы текущего слова, но и предыдущий текст. Кроме того, iTap может предугадывать даже короткие фразы. Такая особенность позволяет существенно ускорить набор текста, особенно если в тексте в основном используются простые и наиболее употребляемые слова и фразы [2].

В общем виде механизм исправления опечаток основывается на двух моделях: модель ошибок и языковая В качестве модели ошибок обычно выступает либо редакционное расстояние, либо модель Бриля-Мура, которая работает на вероятностях переходов одной строки в другую.

Алгоритмическая часть

Для реализации алгоритма мы выбрали язык программирования Kotlin .Основой нашего алгоритма является алгоритм поиска редакционных расстояний или алгоритм Левенштейна.

Редакционное расстояние - минимальное количество операций удаления, вставки или замены символа, необходимое для преобразования одной строки в другую. В качестве строк, будем рассматривать введенное слово и слово из словаря.

Оставим рекуррентную формулу вычисления математикам, алгоритмически вычисления расстояния Левенштейна таково:

D(0, 0) = 0, для всех j от 1 до N включительно;

D(0, j) = D(0, j - 1) + цена вставки символа S2[j], для всех i от 1 до M включительно;

D(i, 0) = D(i - 1, 0) + цена удаления символа S1[i], для всех j от 1 до N включительно;

D(i, j) = минимум из{D(i - 1, j) + цена удаления символа S1[i], D(i, j - 1) + цена вставки символа S2[j],D(i - 1, j - 1) + цена замены символа S1[i] на символ S2[j]};

Где S1 и S2 – сравниваемые строки, длинной m и n соответственно, а D(S1, S2) и есть то самое расстояние. Проще всего представить себе реализацию этого алгоритма через заполнение матрицы m на n. Цены вставки, замены и удаления символов равны единице.

Для применения данного алгоритма мы создали пользовательскую клавиатуру на Android. Ниже указан алгоритм ее написания:

Объявление компоненты IME в манифесте

В системе Android IME - это приложение Android, которое содержит специальную службу IME. Файл манифеста приложения должен объявлять службу, запрашивать необходимые разрешения, предоставлять фильтр намерений, соответствующий действию action.view.InputMethod, и предоставлять метаданные, определяющие характеристики IME. В следующем фрагменте кода объявляется служба IME. Он запрашивает разрешение BIND_INPUT_METHOD, позволяющее службе подключать IME к системе, настраивает фильтр намерений, соответствующий действию android.view.InputMethod, и определяет метаданные для IME:

```
<!-- Declares the input method service -->
<service android:name="FastInputIME"
    android:label="@string/fast_input_label"
    android:permission="android.permission.BIND_INPUT_METHOD">
    <intent-filter>
        <action android:name="android.view.InputMethod" />
    </intent-filter>
    <meta-data android:name="android.view.im"
            android:resource="@xml/method" />
</service>
```

Ниже объявляется действие настроек для IME. У него есть фильтр намерений ACTION_MAIN, указывающий, что это действие является основной точкой входа для приложения IME:

```
<!-- Optional: an activity for controlling the IME settings -->
<activity android:name="FastInputIMESettings"
```

```
        android:label="@string/fast_input_settings">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
        </intent-filter>
    </activity>
```

API метода ввода

Классы, специфичные для IMEs, найдены в android.inputmethodservice и android.view.inputmethod пакетах. KeyEvent класс имеет важное значение для обработки символов клавиатуры.

Центральная часть IME - это сервисный компонент, расширяющийся класс InputMethodService. В дополнение к реализации обычного жизненного цикла службы этот класс имеет обратные вызовы для предоставления пользовательского интерфейса вашего IME, обработки пользовательского ввода и доставки текста в поле, которое в настоящее время находится в фокусе. По умолчанию InputMethodService класс обеспечивает большую часть реализации для управления состоянием и видимостью IME и взаимодействия с текущим полем ввода.

Также важны следующие классы:

BaseInputConnection

Определяет канал связи от InputMethod обратной стороны к приложению, получающему входные данные. Вы используете его для чтения текста вокруг курсора, фиксации текста в текстовом поле и отправки необработанных ключевых событий в приложение. Приложения должны расширять этот класс, а не реализовывать базовый интерфейс InputConnection.

KeyboardView

Его расширение View отображает клавиатуру и реагирует на события ввода пользователя. Раскладка клавиатуры определяется экземпляром Keyboard, который можно определить в файле XML.

Просмотр ввода

Представление ввода - это пользовательский интерфейс, в котором пользователь вводит текст в виде щелчков клавиш, рукописного ввода или жестов. Когда IME отображается впервые, система вызывает onCreateInputView() обратный вызов. В нашей реализации этого метода мы создаем макет, который хотим отобразить в окне IME, и возвращаем его в систему. Этот фрагмент - пример реализации onCreateInputView() метода:

```
override fun onCreateInputView(): View {
    return layoutInflater.inflate(R.layout.input, null).apply {
        if (this is MyKeyboardView) {
            setOnKeyboardActionListener(this@MyInputMethod)
            keyboard = latinKeyboard
        }
    }
}
```

Заключение

В ходе работы были выполнены поставленные задачи: изучены методы компьютерной лингвистики автоматической корректировки правописания, изучена история предикативных систем и общий метод их работы, реализован алгоритма автоматического корректировщика орфографии и разработан интерфейс к ней.

Исправление ошибок и опечаток в словах — непростая задача, особенно для компьютера. Зачастую исправление зависит от согласования форм слов, контекста фразы или целого текста, что усложняет задачу. Однако, как мы убедились, используя разные оптимизации, словари и анализируя соседние и часто используемые слова, компьютер может корректировать орфографию в тексте.

Список используемых источников

- [2] iTap [Электронный ресурс] — Википедия — свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/ITap
- [1] T9 [Электронный ресурс] — Википедия — свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/T9