MAKTABGACHA VA MAKTAB TA'LIMI VAZIRLIGI

MAKTABGACHA VA MAKTAB TA'LIMI VAZIRLIGI

A.AVLONIY NOMIDAGI ILMIY-TADQIQOT INSTITUTI

JIZZAX VILOYATI PEDAGOGIKA MARKAZI

"INNOVATSION TEXNOLOGIYALAR ASOSIDA FAN, TA'LIM VA ISHLAB CHIQARISH INTEGRATSIYASINI TA'MINLASH: MUAMMO VA YECHIMLAR"
XALQARO ILMIY-AMALIY ONLAYN KONFERENSIYASI
(2024-YIL, 15-IYUN)

MATERIALLARI

"ENSURING THE INTEGRATION OF SCIENCE, EDUCATION AND PRODUCTION BASED ON INNOVATIVE TECHNOLOGIES: PROBLEMS AND SOLUTIONS"
INTERNATIONAL SCIENTIFIC AND PRACTICAL ONLINE CONFERENCE
(JUNE 15, 2024 Y)

MATERIALS

РАСПОЗНАВАНИЕ СИМВОЛОВ МЕТОДОМ НАИМЕНЬШЕГО РАССТОЯНИЯ ЛЕВЕНШТЕЙНА

Бекмуратов Косим Аллабердиевич
доцент Самаркандский филиал
ТАТУ им. М. Ал-Хоразми, Узбекистан
+998973952058
bekmurodov1958@mail.ru
Олижонова Саодат Гуломжон кизи
ассистент Самаркандский филиал
ТАТУ им. М. Ал-Хоразми, Узбекистан
+998995996630
solimjonova95@mail.ru

Аннотация Распознавание текста на изображениях является важной задачей машинного обучения, так как позволяет организовать удобное взаимодействие с данными: редактирование, анализ, поиск слов или словосочетаний и т. д. В последние десятилетия благодаря использованию современных достижений компьютерной техники были разработаны новые методы обработки изображений и распознавания образов, что позволило создать такие промышленные системы распознавания текста, как, например, FineReader, удовлетворяющие основным требованиям систем автоматизации документооборота. Однако создание приложения в этой области по-прежнему является творческой задачей и требует дополнительных исследований в связи со специфическими требованиями к разрешению, скорости, надежности распознавания и объему памяти, которые характеризуют каждую конкретную задачу.[1]

Ключевые слова: Распознавание символов, расстояние Левенштейна, редакционное расстояние, обработка текста, оптическое распознавание символов (OCR), исправление ошибок ввода, динамическое программирование, нормализация текста, сегментация символов, шаблонное сопоставление, алгоритмы сравнения строк, редакционные операции, сходство строк, преобразование строк, коррекция опечаток.

Введение

В последнее время задача распознавания символов в прикладных программах не представляет особой сложности - можно использовать множество готовых OCR-библиотек, многие из которых доведены почти до совершенства. Но все же иногда может возникнуть задача разработать собственный алгоритм распознавания без использования сторонних «навороченных» OCR-библиотек.[2]

Именно эта задача возникла у меня в процессе работы, и есть несколько причин, по которым лучше не использовать готовые библиотеки: проект закрытый, с его дальнейшей сертификацией, некий лимит на количество строк код и размер подключаемых библиотек, тем более, что достаточно распознать по предметной области определенный набор символов.

Метод решения задачи:

Алгоритм распознавания прост, и, конечно, не претендует на звание самого точного, быстрого и эффективного, но со своей небольшой задачей справляется хорошо.

Допустим, у нас есть входные данные в виде сканированных изображений документов, структурированная форма. Эти документы имеют специальный односимвольный код, расположенный в верхнем левом углу. Наша задача - распознать этот символ, а затем выполнить какие-то действия, например, классифицировать исходный документ по заданным правилам.

Расстояние Левенштейна. Расстояние Левенштейна, также известное как редакционное расстояние, измеряет минимальное количество операций, необходимых для преобразования одной строки в другую. Операции включают вставку, удаление и замену символов.[3,4]

Применение метода Левенштейна в распознавании символов

1. Предобработка данных: перед применением метода Левенштейна данные обычно проходят этапы предобработки, такие как нормализация текста, удаление шума и сегментация символов.

2. Сравнение строк

- Метод Левенштейна используется для сравнения отсканированных символов с эталонными шаблонами. Например, если нужно распознать символы в изображении, каждый символ сравнивается с набором шаблонов, и выбирается тот, для которого расстояние Левенштейна минимально. [5]

3. Исправление ошибок ввода

- В системах автоматического ввода текста метод Левенштейна помогает в исправлении ошибок ввода, предлагая слова с минимальным редакционным расстоянием к введенному пользователем слову.

Обсуждение. Распознавание символов методом наименьшего расстояния Левенштейна представляет собой задачу, где требуется определить схожесть между строками символов путем минимизации количества операций вставки, удаления и замены символов. Этот метод имеет широкое применение в различных областях, включая оптическое распознавание символов (OCR), исправление ошибок ввода, и обработку естественного языка (NLP). В данной секции обсудим основные аспекты этой задачи, её реализацию и возможные улучшения. [6]

Основные аспекты задачи

1. Сбор и подготовка данных

- Источники данных: В задачах распознавания символов данные могут быть собраны из отсканированных документов, изображений текста, рукописных записей и т.д.[6]

- Предобработка данных: Включает нормализацию текста, удаление шума, бинаризацию изображений, сегментацию символов и выделение контуров символов.

2. Применение расстояния Левенштейна

- Сравнение строк: для каждого отсканированного символа вычисляется расстояние Левенштейна по отношению к эталонным символам. Эталонные символы могут быть предварительно обученными шаблонами.[7]

- Выбор наименьшего расстояния: Символ с минимальным значением расстояния Левенштейна выбирается как распознанный символ.

Преимущества и недостатки метода

Преимущества:

- Простота и эффективность: Метод наименьшего расстояния Левенштейна прост в реализации и эффективно решает задачи сравнения строк.

- Гибкость: Способен обрабатывать различные виды данных и учитывает различные типы ошибок (вставка, удаление, замена).

- Применимость: Широко применяется в OCR, исправлении ошибок ввода и других задачах обработки текста.

Недостатки:

- Вычеслительная сложность: Для больших строк и объемов данных метод может требовать значительных вычислительных ресурсов.

- Чувствительность к шуму: При наличии значительных искажений или шума в данных метод может терять точность.

2. Оптимизация алгоритма

- Эвристики и отсечения: Использование эвристических методов для сокращения пространства поиска и улучшения производительности.

- Параллельные вычисления: Применение параллельных алгоритмов для ускорения вычислений, особенно при обработке больших объемов данных.

3. Интеграция с машинным обучением. Гибридные модели: Комбинация метода Левенштейна с моделями машинного обучения (например, нейронными сетями) для улучшения точности распознавания.

- Обучение на данных: Использование обучаемых моделей для адаптации к конкретным задачам и улучшения качества распознавания.

Практические приложения

1. Оптическое распознавание символов (OCR)

- Применение метода для распознавания текста в отсканированных документах, изображениях и рукописных текстах. Включает обработку различных шрифтов и стилей написания.

2. Исправление ошибок ввода

- Автоматическая коррекция опечаток и ошибок ввода на основе минимального редакционного расстояния. Используется в текстовых редакторах, поисковых системах и мобильных устройствах.

3. Обработка естественного языка (NLP)

- Задачи семантического анализа, кластеризации текстов, машинного перевода и другие приложения, требующие анализа текстовых данных.

Результать. Алгоритм Левенштейна

Алгоритм вычисления расстояния Левенштейна можно описать следующим образом:

1. Инициализация матрицы

- Создается матрица размером (m+1)x(n+1), где m и n—длины сравниваемых строк.

2. Заполнение матрицы

- Заполняются начальные значения: d(i, 0) = i и d (0, j) = j.

3. Динамическое программирование

- Для каждого элемента матрицы рассчитывается минимальное значение среди операций вставки, удаления и замены, добавляя соответствующую стоимость операции.

4. Вывод результата

- Результат (расстояние Левенштейна) находится в правом нижнем углу матрицы.

Пример применения

Рассмотрим пример использования метода Левенштейна для сравнения двух строк: «kitten» и «sitting».

1. Инициализация матрицы

k i t t e n
s
i
t
t
i
n
g

2. Заполнение начальных значений

0 1 2 3 4 5 6
1
2
3
4
5
6
7

3. Заполнение матрицы динамическим программированием

0 1 2 3 4 5 6
1 1 1 2 3 4 5 6
2 2 1 2 3 4 4 5
3 3 2 1 2 3 4 5
4 4 3 2 1 2 3 4
5 5 4 3 2 2 3 4
6 6 5 4 3 3 2 3
7 7 6 5 4 4 3 2

4. Результат. Расстояние Левенштейна между строками «kitten» и «sitting» равно 3.

Вывод