Московский государственный университет имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра математических методов прогнозирования

Шолохова Татьяна Николаевна

Сегментация и анализ изображений древних рукописных текстов

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:

д.т.н., профессор

Л. М. Местецкий

Москва, 2017

Содержание

1 Введение **3**

1.1 Непрерывно-морфологический подход и его применение к распознаванию символов 4

1.2 Постановка задачи 6

2 Основные методы и алгоритмы **7**

2.1 Построение скелета текста 7

2.2 Штриховая сегментация 7

2.3 Сегментация символов 12

2.4 Генерация признаков 13

3 Вычислительные эксперименты **14**

3.1 Исходные данные и условия эксперимента 14

3.2 Алгоритм классификации 14

3.3 Алгоритм кластеризации 16

3.4 Выводы 17

4 Программная реализация **18**

5 Заключение **19**

Список литературы **21**

Аннотация

В работе рассмотрен непрерывно-морфологический подход к задаче оптического распознавания символов. Непрерывно-морфологический подход подразумевает, что основным объектом обработки является не изображение текста, а его непрерывное представление — скелет. Разработаны методы предобработки скелета, сегментирования символов и генерации признакового описания символов. Методы были протестированы алгоритмами классификации и кластеризации, эксперименты показали, что полученное признаковое описание хорошо объясняет геометрию символов.

1 Введение

В современном мире существует потребность преобразования человеческих знаний в электронный вид. Такая необходимость возникла из-за огромного количества проблем, связанных с хранением материальных форм информации: отсутствие возможности быстрого доступа, сложность создания копий, неизбежные повреждения с течением времени. В данной работе рассматривается задача автоматизации процесса оцифровки текстовых данных, называемая оптическим распознаванием символов (optical character recognition, OCR).

Задача оптического распознавания символов частично решена в ряде программных средств и систем (наиболее известны ABBYY FineReader, Tesseract OCR и др). Однако перечисленные программные средства являются коммерческими, поэтому алгоритмы и методы, применяемые в них для решения задач, известны только разработчикам. С эволюцией архитектуры ЭВМ и алгоритмов машинного обучения, её решения активно развиваются. Для различных алфавитов и иногда шрифтов строятся индивидуальные алгоритмы распознавания, для хорошей работы которых необходимы чёткие, неискажённые изображения текста. На сегодняшний день лучшая точность получена для латинских символов и составляет свыше 99%, но абсолютная точность может быть достигнута только при последующем редактировании текста человеком.

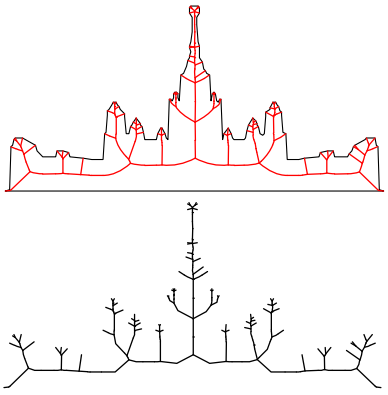
Для многих видов документов ещё не разработан достаточно эффективный алгоритм распознавания, в том числе для тибетских манускриптов. Древние тексты — это не типографская печать. Для них характерны особенности почерка, каллиграфии, носителя (бумаги или ткани) и т.п. Для тибетского письма характерна выраженная каноническая каллиграфия и относительно небольшой алфавит, что дает надежду на успешное автоматическое распознавание.

Необходимость алгоритма распознавания для тибетского языка связана с появлением большого архива изображений тибетских манускриптов, тексты которых содержат интересные для различных наук идеи, методы, точки зрения. В данной работе будет рассмотрен непрерывно-морфологический подход к оптическому распознаванию символов, который позже будет применён к архивам тибетских рукописей.

1.1 Непрерывно-морфологический подход и его применение к распознаванию символов

Основным понятием непрерывно-морфологического подхода является понятие формы объекта; а основной задачей — разработка эффективного способа описания формы, удобного для анализа, распознавания и сравнения форм. Данное понятие позволяет переходить от дискретных терминов (бинарные изображение, пиксели) к непрерывным (фигуры, скелеты фигур). При успешном решении задачи непрерывно-морфологического подхода, появляется возможность создания алгоритмов распознавания наиболее близких человеку. Ведь человек для решения повседневных задач распознавания использует скорее интуитивное понятие формы объекта, чем понятие матрицы пикселей. Наиболее полное описание непрерывно-морфологического подхода можно найти в книге [1].

Определение 1.1. Многоугольной фигурой *M называется связная замкнутая область на плоскости, ограниченная конечным числом отрезков. Соответственно множество всех граничных отрезков фигуры является* **границей фигуры** *∂M.*



(a) Многоугольная фигура и её скелет

(b) Текст и выделенные штрихи

Далее под термином “фигура” понимается именно многоугольная фигура.

Определение 1.2. Скелет — *геометрическое место точек-центров всех вписанных кругов фигуры (то есть имеющих, по крайней мере, две ближайшие точки на границе фигуры). В случае многоугольной фигуры представим множеством прямолинейных и параболических сегментов.*

В работе [2] описан метод построения скелета фигуры и способ его хранения в памяти компьютера: производится аппроксимация исходной фигуры[1] многоугольником; для множества его вершин и отрезков строится диаграмма Вороного, при этом параболические ребра описываются характеристическими многоугольниками (треугольниками) кривых Безье второй степени; из диаграммы Вороного удаляются некоторые отрезки, и получается представление скелета плоским прямолинейным графом $\mathcal{G} := (V, E)$, с максимальной степенью вершины равной 3; параболические ребра имеют очень малую длину по сравнению с кривизной парабол, поэтому для работы со скелетом они приближенно заменяются хордами парабол.

Непрерывно-морфологический подход является удобным и мощным инструментом для распознавания текстов. Понятие скелета позволяет сегментировать основные структурные элементы (например: строки, слова, буквы, штрихи). Штрихом называется естественный элемент письма (например: одно касание пера, соединительная линия). Понятие штриха представляет особый интерес, потому что при качественной сегментации штрихов задача распознавания сводится к перечислению правил написания конкретного текста (например, где и в каком порядке должны находится штрихи некоторой буквы).

В основном для сегментации штрихов в скелете используются эвристические алгоритмы. Например, в работе [1] считается, что штрих представляет собой фрагмент скелетного графа, включающего вершины степени не выше 2. Тогда задача выделения штрихов сводится к разрезанию графа по вершинам степени 3.

После сегментации штрихов возникает необходимость оценить сходство между более общими структурными элементами текста (например, сходство между буквами). В работе [4] предложен способ оценки такого сходства, основанный на ресурсоемком переборе. Сходство между двумя графами здесь определяется количеством неких операций изменения одного графа, которые приводят ко второму.

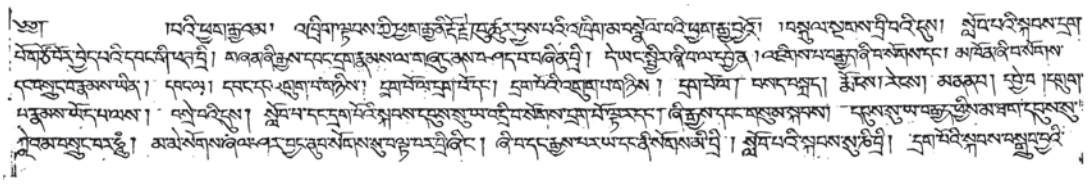
В данной работе предложены другие способы штриховой сегментации и оценки сходства структурных элементов текстов.

---

[1]Исходная фигура состоит из пикселей изображения, до аппроксимации она является многоугольником с прямыми углами и граничными отрезками, параллельными осям координат.

1.2 Постановка задачи

По данным изображениям тибетских манускриптов необходимо построить и реализовать правила извлечения символов тибетского языка. База данных включает изображения размера 7424×1740 пикселей. Все изображения предварительно бинаризованы, избавлены от шума и искажений.



(a) Пример исходного изображения

(b) Буква CA

(c) Результат работы корректного алгоритма, извлекающего букву CA

Необходимо рассмотреть непрерывно-морфологический подход к решению данной задачи: построить и обработать внутренний скелет изображения текста, извлечь из него скелеты символов; классифицировать и агрегировать рёбра скелета в штрихи; выявить признаковое описание символов и протестировать полученное признаковое описание соответствующими эвристическими алгоритмами и алгоритмами машинного обучения; оценить качество полученных методов и алгоритмов.

Так как неизвестны точные решения поставленных задач, критерием оценки является сравнение результатов, полученных предложенными методами, с размеченными вручную ответами.

Рассматриваемый в данной работе подход подразумевает, что правила идентификации основываются на геометрических свойствах письма, а не на растровом представлении символа.

2 Основные методы и алгоритмы

2.1 Построение скелета текста

Определение 2.1. *Скелетом $\mathcal{G}$ текста будем называть объединение скелетов всех связных компонент бинаризованного изображения текста, которые в свою очередь получены алгоритмом описанным выше [2].*

Некоторые обозначения:

- V — множество вершин $\mathcal{G}$. В скелете каждой вершине v соответствуют координаты на плоскости (v_x, v_y). Также каждой вершине можно приписать некоторый номер, тогда v_i обозначает i-ую вершину графа.
- $E \subset V \times V$ — множество рёбер $\mathcal{G}$. Граф неориентированный, поэтому не ограничивая общности можно считать $(v_i, v_j) \in E \implies i < j$.
- $|v|$ — степень вершины v, E_v — множество рёбер, инцидентных вершине v.
- $|e| = |(u, v)|$ — длина ребра, $\angle(e', e'')$ — угол в радианах между инцидентными рёбрами e' и e''.

2.2 Штриховая сегментация

Визуальный анализ изображений текста показывает, что каллиграфия тибетского письма основывается на некотором небольшом наборе штриховых элементов. Причем, множество штрихов меньше, чем множество символов алфавита. Изображения букв формируются из этих элементов. Поэтому в работе используется двухступенчатый подход. Сначала выполняется “штриховая” сегментация. Она состоит в выделении в скелете подграфов, которые можно классифицировать как штрихи. Поскольку эти подграфы формируются “снизу” из цепочек ребер скелета, этот процесс назван “агрегированием” или “агрегацией”. Второй шаг — это работа со штрихами, распознавание букв на основе выделения и классификации групп штрихов.

Первичная агрегация

Использовалась жадная идея: пока в скелете есть рёбра малой длины, либо есть пары

соседних рёбер с близким к развёрнутому углом между ними, рёбра объединяются в одно (происходит удаление некоторых вершин степени 2).

Пусть V' — подмножество вершин графа $\mathcal{G}$, которые будут удалены во время очередной итерации. Рассмотрим алгоритм.

Алгоритм 1. Первичная агрегация

Повторять
  $V' = \varnothing$
  Цикл по вершинам графа $v \in V$ **:**
    Если $|v| = 2$ **тогда**
      $e', e'' = E_v$
      Если $|e'| \leq \varepsilon_{Length}$ **или** $|e''| \leq \varepsilon_{Length}$ **или** $\angle(e', e'') \approx \pi$ **тогда**
        $V' = V' \cup \{v\}$
      конец условия
    конец условия
  конец цикла
  Цикл по удаляемым вершинам $v \in V'$ **:**
    $e', e'' = (u, v), (v, w) = E_v$
    $V = V \setminus \{v\}$
    $E = E \setminus \{e'\}$
    $E = E \setminus \{e''\}$
    $E = E \cup \{(u, w)\}$
  конец цикла
Пока $V' \neq \varnothing$

Направления штрихов

В полученном после агрегации скелете штрихи отличаются между собой положением, длиной и направлением. Для построения признакового описания происходит разделение штрихов по направлениям на четыре типа: диагональные E_{diag}, антидиагональные E_{adiag}, вертикальные E_{vert} и горизонтальные E_{hor}.

Каждому направлению соответствует шаблонный вектор длины 1. Для определения типа очередного штриха, подсчитываются абсолютные величины синусов углов между данным штрихом (штрих (u, v) преобразуется в вектор $(v.x - u.x, v.y - u.y)$) и всеми шаблонными векторами. Т.к. $|\sin\angle(\vec{a}, \vec{b})|$ является показателем близости направлений векторов $\vec{a}$ и $\vec{b}$. Итоговым типом штриха является тип шаблона, оказавшийся ближайшим. Рассмотрим алгоритм.

Алгоритм 2. Вычисление направлений штрихов

$\vec{s}_{diag} = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$
$\vec{s}_{adiag} = (-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$
$\vec{s}_{hor} = (1, 0)$
$\vec{s}_{vert} = (1, 0)$
Цикл по рёбрам графа $e = (u, v) \in E$ **:**
$\vec{s} = (v.x - u.x, v.y - u.y)$
$OptimalDir = \underset{dir \in \{diag, adiag, hor, vert\}}{\arg\min} |\sin \angle(\vec{s}_{dir}, \vec{s})|$
$E_{OptimalDir} = E_{OptimalDir} \cup \{e\}$
конец цикла

Рис. 3: Скелет до и после первичной агрегации с выделенными направлениями штрихов

Вторичная агрегация

Первичная агрегация плохо работает в случае закруглённых областей скелета (например, на наборе из большого количества коротких штрихов, которые на самом деле не соответствуют прямой линии Рис. 3). Поэтому, после первичной необходима вторичная агрегация.

Алгоритм вторичной агрегации устроен следующим образом:

1. весь скелет разбивается на цепочки вершин, цепочка — упорядоченное подмножество вершин исходного графа $\mathcal{G}$, с выделенными начальной и конечной вершинами, все внутренние вершины которого имеют степень 2;
   $C_k = (c^k_{(1)}, c^k_{(2)}, \ldots, c^k_{(n)}), c_{(i)} \in V, |c_{(i)}| = 2 \quad \forall i = \overline{2, n-1}, |C_k| = n$ — количество вершин в цепочке, $V = \bigcup\limits_k C_k$.
2. для каждой цепочки C производится следующий итерационный процесс (i - номер итерации):
   (a) выбирается очередная начальная вершина c^i_0 (для первой итерации — первая вершина цепочки $c_0 = c_{(1)}$, для всех последующих итераций — конечная вершина предыдущей итерации);

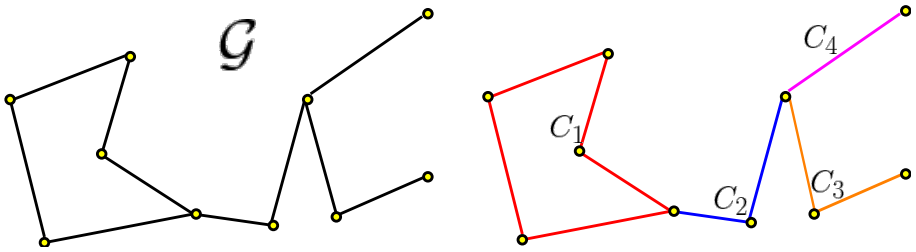
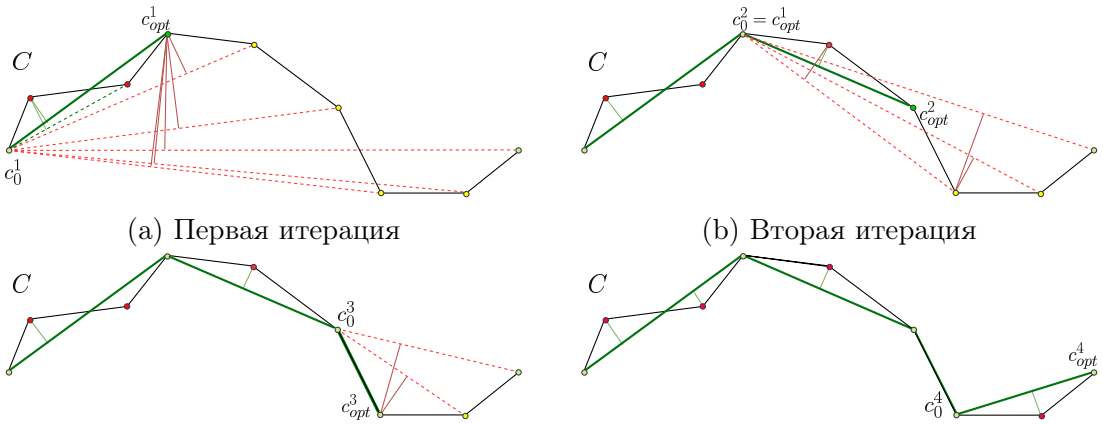


Рис. 4: Пример разбиения графа $\mathcal{G}$ на цепочки C_i

Для цепочки C_1 начальная и конечная вершина совпадают и имеют степень 3.

(b) перебором находится самая дальняя конечная вершина c^i_{opt}, для которой выполняется *оптимальное* условие малости угла: максимальный угол между результирующей прямой и прямой, соединяющей начальную вершину с очередной в цепочке, не должен превышать фиксированного числа. Пусть вектор направления результирующей прямой $\vec{opt} = (c^i_{opt}.x - c^i_0.x, c^i_{opt}.y - c^i_0.y)$, D_i — множество вершин цепочки между начальной и текущей оптимальной вершинами, $\vec{cur}_d$ — вектор между начальной и вершиной d. Тогда данное условие можно записать формально: $\max\limits_{d \in D} \angle(\vec{opt}, \vec{cur}_d) < \varepsilon$.



(a) Первая итерация

(b) Вторая итерация

(c) Третья итерация

(d) Четвёртая итерация

Рис. 5: Иллюстрация работы алгоритма вторичной агрегации для одной цепочки

На рисунке 5 пунктирными линиями обозначены линии, соединяющие начальную вершину итерации с очередной рассматриваемой вершиной цепочки. Зелёная пунктирная линия означает, что условие оптимальности выполнено, красная — не выполнено.

Улучшенная вторичная агрегация

Для этого метода агрегации использовался алгоритм описанный выше, но условие малого угла было заменено на условие малого отклонения от регрессионной прямой: максимальное расстояние между точкой цепочки и прямой с минимальным среднеквадратичным отклонением от всех точек цепочки не превышает выбранной константы.

Нахождение оптимальной прямой подразумевает решение задач регрессии для всевозможных начальных наборов точек цепочки. Задача регрессии формулируется следующим образом: пусть задано множество точек плоскости $\{(x_i, y_i), i = \overline{1,n}\}$, тогда коэффициенты оптимальной прямой $y = ax + b$ находятся из следующего условия $L(a,b) = \sum\limits_{i=1,n} (y_i - (ax_i + b))^2 \to \min\limits_{a,b}$. Для минимизации необходимо подсчитать производные и приравнять их к нулю.

$$\frac{\partial L}{\partial a} = \sum_{i=1,n} -x_i(y_i - (ax_i + b)) = 0; \quad \frac{\partial L}{\partial b} = 2\sum_{i=1,n} (y_i - (ax_i + b)) = 0;$$

Итого получается система следующего вида:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} E \\ F \end{bmatrix}$$

$$A = \sum x_i^2,\ B = \sum x_i y_i,\ C = -\sum x_i,\ D = \sum y_i,\ E = \sum x_i y_i,\ F = -\sum y_i;$$

Методом Крамера получаются решения:

$$\Delta = AD - BC;\ \Delta_a = ED - BF;\ \Delta_b = AF - CE;$$

$$a = \frac{\Delta_a}{\Delta};\ b = \frac{\Delta_b}{\Delta};$$

Так, при добавлении очередной точки цепочки необходимо добавить по одному слагаемому в соответствующие параметры A, B, C, D, E, F и вычислить новые оптимальные коэффициенты прямой a и b.

Когда оптимальная регрессионная прямая вертикальная $x = const$, уравнение не представимо в виде $y = ax + b$. Поэтому, описанным выше способом решалась ещё

одна регрессионная задача $x = a'y + b'$, $\sum\limits_{i=1,n} (x_i - (a'y_i + b'))^2 \to \min\limits_{a',b'}$. После этого выбирается прямая, с наименьшим среднеквадратичным отклонением.

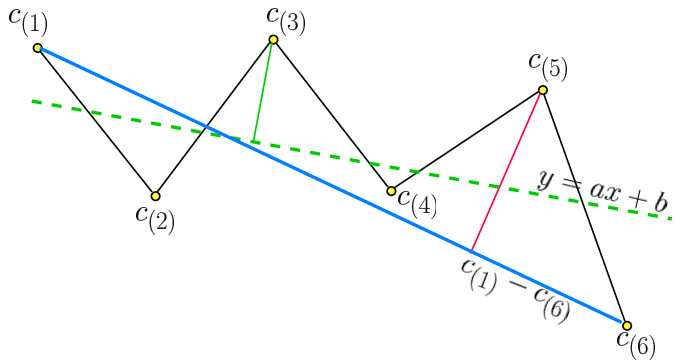


Рис. 6: Иллюстрация работы алгоритма улучшенной вторичной агрегации

На рисунке 6 голубым цветом обозначена линия, оптимальная с точки зрения вторичной агрегации, а зелёным пунктиром — прямая, оптимальная с точки зрения улучшенной вторичной агрегации. Пример демонстрирует случай, когда улучшенная вторичная агрегации сглаживает цепочку, а обычная вторичная агрегация не сглаживает.

2.3 Сегментация символов

Для сегментации символов используется эвристический подход. В случае данного подхода для каждой буквы необходимы индивидуальные правила идентификации. Это возможно, поскольку алфавит небольшой.

Например: на всех изображениях размер буквы CA почти не изменяется, поэтому для выделения очередной буквы используется рамка фиксированного размера $2 \cdot d$; дополнительно, в букве CA всегда есть длинная диагональ по центру символа.

Таким образом получается следующий алгоритм выделения букв (результатом алгоритма является граф $G' = (V', E')$ — подграф исходного графа $\mathcal{G}$):

1. рассматриваются все диагональные штрихи $e = (u, v) \in E_{diag}$;
2. строится квадратная рамка R, центр которой совпадает с центром диагонального штриха $x = \frac{u_x + v_x}{2}$, $y = \frac{u_y + v_y}{2}$, $R = \{[x - d, x + d] \times [y - d, y + d]\}$;
3. находятся все вершины V', находящиеся внутри рамки R, $V' = \{v | v \in V, (v_x, v_y) \in R\}$;
4. восстанавливаются рёбра между внутренними вершинами $E' = \{(u, v) | (u, v) \in E, u \in V', v \in V'\}$.

Полученный подграф G' считается очередной буквой, которую необходимо идентифицировать.

Кроме буквы CA под описание, данное выше подходят некоторые другие буквы тибетского языка, а также части букв, содержащих в себе диагональный элемент.

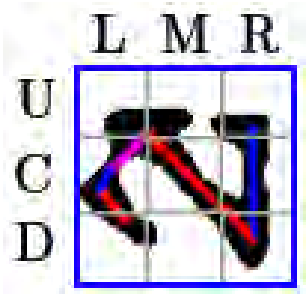
Рис. 7: Примеры выделенных рамок

2.4 Генерация признаков

Для каждой буквы, генерируются признаки, на которых основана дальнейшая идентификация. Положение штриха внутри рамки является важным признаком. Рамка равномерно разделяется на 9 секторов (LU, LC, LD, MU, MC, MD, RU, RC, RD), штриху в соответствие ставился тот сектор, в который попадает центр данного штриха (Рис. 8). Также штрихи делятся на короткие(1), средней длины(2) и длинные(3) по некоторым порогам.

Итого получается $4 \times 9 \times 3 = 108$ возможных типов штрихов. В качестве признаков используются гистограммы распределения штрихов по типам, где все штрихи вносят одинаковый вес и аналогичные гистограммы, где каждый штрих вносит вес, пропорциональный своей длине.

Дополнительно для всего подграфа-символа подсчитываются некотрые характеристики, такие как: длина минимального и максимального рёбер, самый длинный путь(диаметр подграфа), средняя длина всех путей, количество компонент связности и некоторые функции от данных величин. Данные характеристики вычисляются с помощью подсчета попарных расстояний между вершинами графа алгоритмом Флойда-Уоршелла [5].



Пример

Подграф, изображённый на рисунке 8 содержит рёбра следующих типов: LC.adiag.2 (левый центральный блок, антидиагональное, короткое), LC.vert.1, LC.diag.1, MC.diag.2, RD.diag.1, RC.vert.2.

Рис. 8: Признаковое описание

3 Вычислительные эксперименты

3.1 Исходные данные и условия эксперимента

Эксперименты проводятся на данной базе изображений (пример 2а). Для алгоритма классификации данные размечались следующим образом: вручную размечались точки изображения, соответствующие центрам буквы СА. Объектами являлись рамки, центр которых совпадал с центром некоторого диагонального штриха. Объект считался принадлежащим классу СА, если центр его рамки находился достаточно близко к одной из размеченных точек. Всего было размечено 10 изображений, содержащих в среднем 70 букв СА. Генерировалась выборка по размеченным изображениям и качество алгоритмов классификации оценивалось с помощью кросс-валидации.

3.2 Алгоритм классификации

Эвристические правила идентификации

Эвристические наблюдения в случае буквы СА следующие: по центру рамки всегда находится длинный диагональный штрих; снизу слева встречается диагональный штрих средней длины, но допускается и горизонтальный; справа по центру часто виден вертикальный штрих средней длины и т.п. За каждый штрих, включенный в наблюдения и присутствующий на графе добавлялась некая цена. После этого, буквы, графы которых получили цену превышающую пороговую считались буквами СА.

Динамические правила идентификации

Рис. 9: Примеры скелетов для букв CA

Можно автоматизировать подбор цен для каждого штриха с помощью алгоритма классификации. Для этого необходимо сгенерировать выборку данных. Разметка нескольких изображений проводилась вручную, после чего на описанных выше признаках обучались следующие алгоритмы: логистическая регрессия, метод опорных векторов, случайные леса.

Оценка качества

Так как данные несбалансированны (на одном изображении из всех найденных рамок $\approx 7\%$ на самом деле соответствуют букве CA).

Выбранная оценка качества AUC_PR — площадь под precision recall кривой. Метрика учитывает несбалансированность данных.

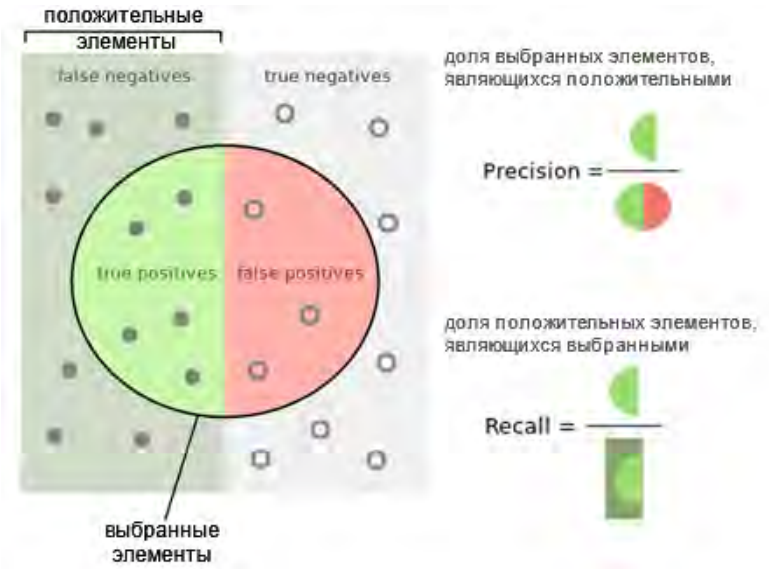


Рис. 10: Precision Recall

Отдельные проблемы связаны, со способом выбора рамок, так как для некоторых букв CA агрегация не объединила длинную центральную диагональ и как следствие в центре этой буквы не будет рассмотрена рамка, такие буквы не войдут в выборку и не будут учтены при подсчёте качества.

Результаты эксперимента

Среднее количество вершин в исходном скелете $|V| \approx 10^5$.

Количество вершин в скелете после первичной агрегации $|V| \approx 2 \cdot 10^4$.

Количество вершин в скелете после вторичной агрегации $\approx 2 \cdot 10^3$. Алгоритм, выполняющийся для каждой цепочки, в худшем случае работает за $\mathbf{O}(|C|^3)$ операций. Но скелет так устроен, что не встречаются цепочки, длина которых превышает 15 звеньев. Итого количество операций $\approx \frac{2 \cdot 10^4}{15} \cdot 15^3 \approx 2 \cdot 10^5$.

Среднее время построение скелета для одного изображения занимает 5-7 секунд, суммарное время работы всех остальных используемых алгоритмов не превышает 5 секунд.

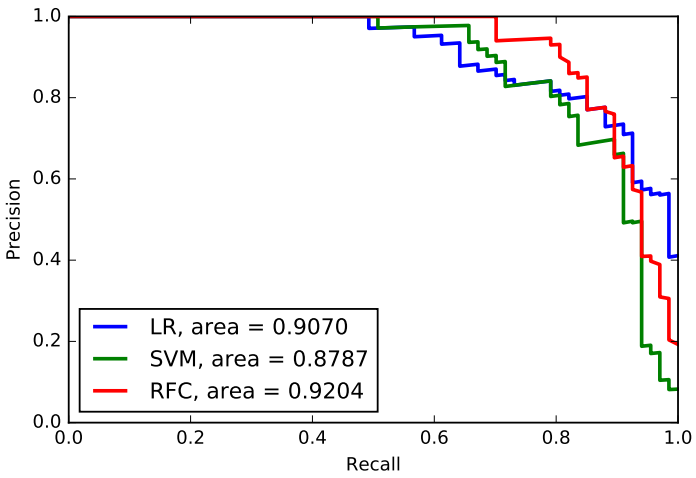


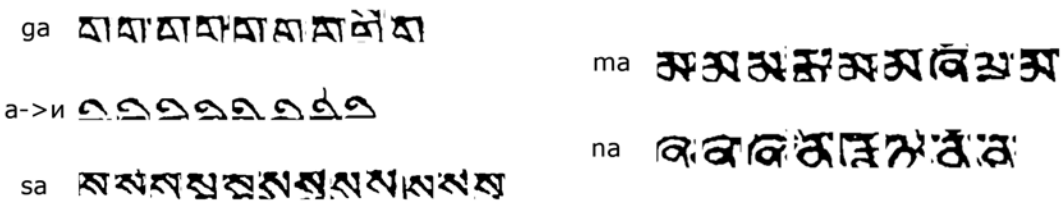
Рис. 11: Precision Recall Curves

Полученные результаты говорят о том, что с наилучшей стороны себя проявил алгоритм случайных деревьев. Невысокая точность остальных алгоритмов оправдывается небольшой обучающей выборкой, неоптимальным выбором центров рамок и высокой скоростью обработки каждого изображения.

3.3 Алгоритм кластеризации

На описанных выше объектах был запущен алгоритмы кластеризации (k-средних). Цель такого эксперимента — продемонстрировать, что полученные признаки объясняют структуру букв.

Для этого производилась кластеризация с 60 кластерами, использовалась стандартная евклидова метрика. В результате (как и предполагалось) получались небольшие кластеры, содержащие либо одинаковые буквы, либо схожие в написании буквы.



(a) Хорошие кластеры (b) Плохие кластеры

Рис. 12: Примеры кластеров

Тибетский алфавит состоит из тридцати основных букв и четырёх гласных. Тридцать основных букв можно назвать согласными, однако нужно учитывать то, что тибетское письмо — слоговое письмо. И каждая буква является не просто буквой, а буквой-слогом, т.е. включает в себя и согласный и гласный звуки (если нет символов указывающих на определённый гласный звук, то это «а»). Эти символы, которые меняют гласный звук «а» в слоге, как раз и есть те самые четыре гласные. На рисунке 12 в примерах хороших кластеров можно увидеть одну из гласных букв.

3.4 Выводы

Рассмотрен непрерывно-морфологический подход к задаче оптического распознавания символов. В самом начале работы выполнен переход от дискретного представления изображения к непрерывному, дальнейшая работа опиралась на основную структуру непрерывного подхода — скелет.

Работу можно разбить на две глобальные части: обработка скелета, выделение и поиск букв в скелете.

Для обработки скелета разработаны два метода агрегации, был реализован специальный модуль написанный на C++. Для выделения и поиска букв, извлечения признаков и подсчёта точностей использовался язык Python.

Для повышения качества предложенного алгоритма необходимо увеличение обучающих данных. Масштабная проблема связана с выбором размеров и центров рамок

символов, её можно решить, улучшив обработку скелета, либо пожертвовав эффективностью алгоритма (рассматривать больше возможных центров рамок).

4 Программная реализация

В ходе выполнения работы написаны модули на языках C++ и Python.

- **Модуль BuildSkel**.

  Реализован на C++ (VS2015). Производит построение скелета и выполняет первичную агрегацию.

  Шаблон вызова:

  `BuildSkel img_in graph_out img_out eps_angle=0.2 eps_len=2`

  Файл `img_in` содержит исходное изображение в формате bmp; `graph_out` — текстовый документ, содержит граф в описанном ниже формате; `img_out` — изображение, визуализация полученного графа в формате bmp; `eps_angle, eps_len` — параметры первичной агрегации.

- **Модуль SecondAgg**.

  Реализован на C++ (VS2015). Производит вторичную агрегацию скелета.

  Шаблон вызова:

  `SecondAgg in out eps=0.2`

  `in` содержит граф после первичной агрегации в описанном ниже формате; `out` содержит граф после вторичной агрегации в аналогичном формате. `eps` — параметр вторичной агрегации.

- **Модуль ThirdAgg**.

  Реализован на Python3.5. Производит улучшенную вторичную агрегацию скелета.

  Шаблон вызова:

  `ThirdAgg in out eps_agg3=1`

  `in` — текстовый документ, содержит граф в описанном формате; `out` — текстовый документ, содержит агрегированный граф в описанном формате; `eps_agg3` — параметр улучшенной вторичной агрегации.

- **Модуль graph**.

  Реализован на Python3.5. Содержит функции, связанные с обработкой и визуализацией скелета, выделением подграфов-символов.

- **Модуль get_samples**.

  Реализован на Python3.5. Содержит функции, генерирующие признаки.

Формат хранения графа:

N — количество вершин графа, далее N строк

...

$n_i \;\; x_i \;\; y_i$ — номер и координаты очередной вершины

...

M — количество рёбер графа, далее M строк

...

$n_{fromj} \;\; n_{toj}$ — пара номеров вершин, задающих очередное ребро

...

5 Заключение

В результате данной работы:

- Исследован непрерывно-морфологический подход к задаче оптического распознавания символов.

- Разработаны, реализованы и протестированы два метода сегментации штрихов (структурная единица письма, например, касание пера) в скелете текста: эвристический и переборный.

- Разработан и реализован метод генерации признакового описания подграфов-символов. На полученных признаках обучено несколько алгоритмов классификации и кластеризации.

- Проведены вычислительные эксперименты, которые показали работоспособность разработанных методов.

Возможным улучшением данного метода является более детальное исследование положений символов тибетского языка, например, с помощью внешнего скелета можно сегментировать строки текста, а по положениям строк находить центры символов, это избавит от необходимости генерировать правила извлечения для каждого символа в отдельности.

Другим улучшением может служить отказ от предположения о том, что символ содержится в рамке, фиксированного размера. Триангуляция Делоне позволяет находить близко расположенные штрихи (не обязательно инцедентные). Далее по каждому штриху и его ближайшим можно генерировать признаковое описание.

В данной работе в ходе агрегации вершины могли только удаляться, но не могли менять свои координаты. Возможно отказаться от работы со скелетом, как с графом, а работать со скелетом как с множеством рёбер. Тогда положения вершин смогут меняться, что, возможно, позволит получить более качественную сегментацию штрихов (например, в ходе улучшенной вторичной агрегации можно проецировать края цепочки на регрессионную прямую).

Список литературы

[1] Местецкий Л.М. “Непрерывная морфология бинарных изображений: фигуры, скелеты, циркуляры.” // Москва, Физматлит, 2009.

[2] Местецкий Л.М. “Скелет многоугольной фигуры – представление плоским прямолинейным графом.” // ГРАФИКОН-2010, Санкт-Петербург, 2010.

[3] Шолохова Т.Н. “Метод распознавания символов в изображениях тибетских манускриптов” // Сборник тезисов XXIV Международной научной конференции студентов, аспирантов и молодых ученых «Ломоносов-2017» секция «Вычислительная математика и кибернетика». Издательский отдел факультета вычислительной математики и кибернетики МГУ имени М. В. Ломоносова, 2017.

[4] Xinbo Gao, Bing Xiao, Dacheng Tao, Xuelong Li “A survey of graph edit distance.” // FORMAL PATTERN ANALYSIS&APPLICATIONS, 2010.

[5] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. “Алгоритмы: построение и анализ.” // МЦНМО, 2000.