МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ)

Физтех-школа прикладной математики и информатики (ФПМИ)
Факультет инноваций и высоких технологий (ФИВТ)
Кафедра когнитивных технологий

СЕГМЕНТАЦИЯ БИНАРНЫХ ТЕКСТОВЫХ ДОКУМЕНТОВ

Отчёт о выполненной работе

Выполнил: Сухарников А.А.
Проверил: Полевой Д.В.
Группа Б05-812

Москва 2021

Содержание

1. Введение 3

2. Описание решения 3

3. Пользовательское описание 4

4. Тестовый датасет 5

5. Сборка и развёртывание 6

6. Результаты 7

Список литературы 8

Рис. 1: Пример разметки страницы

Рис. 2: MST на компонентах связности

1. Введение

Данная задача заключается в извлечении из изображений страниц текстового документа однородных компонентов: участков с текстом, таблиц, графиков, рисунков, отдельных строк (заголовков) и т.д. [1]. При этом она не включает классификацию обнаруженных компонент на перечисленные выше типы, а требует только нахождения их областей на изображении.

Опишем решаемую в данной работе задачу более формально. На вход поступает бинарный текстовый документ (имеющий только два возможных цвета – чёрный и белый). При этом предполагается именно печатный текст, а не рукописный (в частности, чтобы различные его компоненты не пересекались). На выходе генерируется набор изотетических полигонов – т.е. фигур, имеющих только вертикально или только горизонтально направленные рёбра [2]. Каждая такая фигура определяет какую-то отдельную компоненту документа – участок текста, таблицу, рисунок и т.п. Пример ожидаемого результата работы изображён на рис. 1.

2. Описание решения

В качестве решения задачи был применён подход с использованием минимального остовного дерева (Minimal Spanning Tree, или просто MST). Изначально, на исходном бинаризованном документе производится поиск компонент связности. Компонента связности в контексте изображения определяется так: пусть пиксели изображения будут вершинами графа, и каждая из этих вершин будет вязана с восемью соседними пикселями-вершинами рёбрами, тогда компонента связности такого графа и есть компонента связности изображения. Например, это буквы печатного документа. Затем найденные компоненты связности, в свою очередь, сами рассматриваются как вершины, и на этих вершинах строится MST (алгоритмом Крускала в данной реализации, рис. 2).

Далее происходит первое разделение на компоненты документа путём удаления части рёбер из построенного MST. Ребро удаляется, если оно меньше среднего размера компоненты связности изображения с некоторым заранее заданным коэффициентом. Однако, как выяснилось из экспериментов, такого подхода зачастую недостаточно для разделения участков текста на абзацы (т.к. обычно расстояния между ними такие же, как и

между строками в абзаце), поэтому была добавлена дополнительная обработка. В каждой полученной ранее (текстовой) компоненте удаляются вертикальные рёбра: тем самым выделяются компоненты-строки, для которых затем вычисляется bounding box (минимальный прямоугольник, содержащий граф/подграф – строку в данном случае). В некоторых случаях истинная строка документа по ошибке может получиться разбитой на несколько bounding box'ов, поэтому предусмотрено объединение bounding box'ов, располагающихся примерно в одной строке. На этом этапе каждой строке соответствует один bounding box. Затем, ищутся смежные (по вертикали, других не будет) bounding box'ы, имеющие смещение, характерное для начала абзаца. Соответственно, такие bounding box'ы считаются принадлежащими различным абзацам, и компонента разделяется на две по смежной границе этих box'ов. После этого разделения строки-box'ы, опознанные как принадлежащие одной абзацу/компоненте документа, объединяются в изотетические полигоны, которые были описаны ранее.

3. Пользовательское описание

Итоговое решение представляет собой набор из трёх утилит: утилита парсинга XML-файлов датасета, утилита сегментации документа, утилита оценки качества.

Утилита парсинга XML-файлов *xml_parser.exe* нужна для извлечения координат расположения компонент. Файлы датасета содержат большое количество ненужной для оценки данной задачи информации – тип компоненты, ориентацию, направление чтения текста и т.п. Для задачи сегментации же нужны только описание расположения компонент, что и извлекает эта утилита. Принимает два аргумента:

1) путь к файлу XML датасета

2) директория, куда сохранить итоговый файл разметки (только расположение компонент).

Применение данной утилиты не является обязательным – она нужна только для генерации истинной разметки корректного формата. Пример вызова:

xml_parser.exe data/xml/example1_263.xml data/images/example1_263.txt

Утилита сегментации документа *segmentation.exe* по переданному ей изображению страницы возвращает найденную разметку документа в виде отдельного файла (в формате, понятном утилите оценки качества), страницу с отрисованной поверх неё разметкой и страницу с отрисованным поверх MST (иллюстрация к деталям реализации). Принимает два аргумента:

1) путь к исходному изображению страницы (не обязательно бинарному, бинаризуется автоматически)

2) директория, куда сохранить ответ (три файла: разметка, изображение с разметкой синим цветом и изображение с MST).

Пример вызова:

segmentation.exe data/images/example1_263.tif out

Утилита оценки качества *evaluation.exe* по передаваемым на вход истинной и найденной разметкам выводит значения различных типов ошибок сегментации. Принимает четыре аргумента:

1) путь к исходному изображению страницы
2) путь к файлу найденной разметки (полученной от *segmentation.exe*)
3) путь к файлу истинной разметки (полученной от *xml_parser.exe*)
4) директория, куда сохранить ответ (изображение страницы с истинной разметкой красным; изображение с отрисованной на ней истинной и найденной разметками, красным и синим цветами соответственно).

Пример вызова:

evaluation.exe data/images/example1_263.tif out data/test/example1_263.txt out

4. Тестовый датасет

Для проверки качества сегментации взят PRImA Layout Analysis dataset (описание [2] и сам датасет [3], по ссылке таже есть веб-интерфейс для просмотра сэмплов датасета). Он содержит порядка 400 страниц статей научного и научно-популярного характера. Встречаются страницы различных типов (как манхэттенские, т.е. границы компонент которых только вертикальны или горизонтальны, так и неманхэттенские), с рисунками, таблицами, графиками. К каждому документу приведена истинная разметка в виде XML файла, содержащего для каждой его компоненты последовательность его вершин (в порядке обхода по рёбрам). Для тестирования отобрано несколько примеров. Вообще, датасет по своему назначению предназначен для оценки результатов решения более широкой задачи – помимо сегментации страницы, также классификации найденных компонент (на абзацы, таблицы, рисунки и т.п.). Поэтому инструмент оценки качества, прилагающийся к данному датасету, применить исключительно к задаче сегментации не удалось. Вместо него был реализован отдельный модуль оценки качества в самой программе, реализующей алгоритм сегментации. Далее опишем подход, применённый в этом модуле.

Оценка осуществляется способом, приведённым в описании датасета [4]. Итоговая ошибка складывается из пяти различных видов ошибок: False Detection, Miss, Partial Miss, Split и Merge. False Detection – алгоритмом найдена компонента, которая не пересекается ни с одной из компонент истинной разметки. Miss – обратная ситуация, компонента истинной разметки не пересекается ни с одной из найденных компонент. Partial Miss – есть частичное пересечение. Split – несколько найденных компонент пересеклись с одной компонентой истинной разметки. Merger – наоборот, несколько компонент истинной разметки пересеклись с одной найденной компонентой. Ошибка каждого типа подсчитывается как

доля площади изображения, содержащей ошибку (для Partial Miss, например, это доля непокрытой истинной разметки + доля непокрытой найденной алгоритмом разметки).

5. Сборка и развёртывание

- Установить cmake версии не ниже 3.10. Установить OpenCV (рекомендуется 4.5.2, т.к. тестирование проводилось именно с ней), определить путь к директории с билдом OpenCV. Установить Doxygen (рекомендуется 1.9.3).

- Скачать проект с репозитория:

  `https://github.com/and15121512/sem-8-page-segmentation`.

  Также можно догрузить несколько изображений датасета с гугл диска и протестировать программу на них (изображения не влезли в допустимый размер репозитория, истинная разметка для них уже в репозитории):

  `https://drive.google.com/drive/folders/1uzjmYrA7KiYw1xtKDtVs8lmI5G-WMEV9?usp=sharing`.

- Далее, все пути в этой инструкции указываются относительно корня проекта. Создать директорию ./build и перейти в неё:

  ```
  mkdir build
  cd build
  ```

- Выполнить команду:

  ```
  cmake -DOpenCV_DIR=*Путь к дир-рии
      сборки opencv с файлом OpenCVConfig.cmake* ..
  ```

  Если cmake сможет определить сборку OpenCV, в логе отобразится строка с “. . . Found OpenCV . . . “. Для динамической версии библиотеки cmake также дополнительно укажет директорию с .dll, которую необходимо добавить в переменную PATH (перезагрузив затем сессию терминала).

- Выполнить команды сборки и установки проекта:

  ```
  cmake --build . --config Release
  cmake -P cmake_install.cmake
  ```

  В директории ./build/bin/ в результате появятся исполняемый файл программы, копия директории ./data/ с файлами для тестирования (её содержимое описано ранее) и директория .build/bin/out/, в которую впоследствии будет сохранён вывод тестов. Помимо этого, в директории ./build/doc/html будет собрана документация к интерфейсам модулей кода (**Doxygen может отработать некорректно при наличии символов Юникода. Лучше использовать полные пути только на английском!**).

- Выполнить команду для запуска всех тестов:

  ```
  ctest -C Release
  ```

  В результате в консоль будет выведен статус выполнения каждого из тестов. Ожидается, что все они будут пройдены корректно. Также можно визуально оценить качество работы алгоритма, просмотрев изображения с префиксом из директории ./build/bin/out. Синим цветом отмечена найденная разметка.

- Для оценки качества тестовых примеров перейти в директорию ./build/bin выполнить команды (в директории out/ появится истинная разметка, красным цветом, и наложенные найденная и истинная разметки с соответствующими цветами):

evaluation.exe data/images/example1_263.tif out/example1_263.txt data/layout/example1_263.txt out
evaluation.exe data/images/example2_648.tif out/example2_648.txt data/layout/example2_648.txt out
evaluation.exe data/images/example3_674.tif out/example3_674.txt data/layout/example3_674.txt out

6. Результаты

В процессе разработки алгоритма внимание в основном было уделено возможности выделения отдельных абзацев в тексте (в датасете такое разделение производится). В связи с этим алгоритм даёт низкие показатели суммарной ошибки для страниц, содержащих только текстовые данные примерно одного размера шрифта (примеры 1, 2, 9 порядка 0.02%).

Если размеры текста значительно варьируются (есть какой-то крупный текст, например, как на плакатах), качество довольно сильно ухудшается. Это связано с тем, что для стабильной работы алгоритма удалялись большие компоненты связности (буквы). Примеры таких страниц (ошибка 30% в среднем): 5, 10.

Таблицы и рисунки алгоритм детектировать практически не способен: т.к. внимание было сдвинуто в сторону работы с текстовыми компонентами, алгоритм пытается работать с другими типами компонент (заранее классификация неизвестна) как с текстом, а на самом деле структура таблиц и рисунков может довольно сильно отличаться. Это видно по примерам 4, 6, 7, 8 (ошибка порядка 20%).

В итоге, алгоритм достаточно хорошо работает на примерах, содержащих преимущественно текст, при этом в среднем одного размера шрифта (например, научные статьи).

Ex. No	1	2	3	4	5	6	7	8	9	10
Error	0.02	0.02	0.18	0.17	0.37	0.20	0.01	0.21	0.01	0.20

Список литературы

[1] Koichi Kise. Page segmentation techniques in document analysis. *Handbook of Document Image Processing and Recognition*, pages 135–175, 2014. doi:10.1007/978-0-85729-859-1_5.

[2] C. Clausner, A. Antonacopoulos, and S. Pletschacher. Icdar2019 competition on recognition of documents with complex layouts – rdcl2019. *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 2019.

[3] A. Antonacopoulos, D. Bridson, C. Papadopoulos, and Pletschacher S. Prima layout analysis dataset. (Дата обращения: 13.02.2021). URL: https://www.primaresearch.org/dataset/.