Санкт-Петербургский государственный университет

Кафедра системного программирования

Кеворкянц Павел Георгиевич

Разработка программного обеспечения для автоматического перевода изданий комиксов

Отчет по учебной практике

Научный руководитель:
Заведующий кафедрой СП, Терехов А. Н.

Санкт-Петербург 2021

Оглавление

Введение

Ежегодно, преимущественно в Соединенных Штатах Америки, производится большое количество изданий самых различных комиксов. Каждое такое издание российское комьюнити любителей комиксов старается как можно быстрее адаптировать и перевести на русский язык. Однако, этот процесс может растягиваться на длительное время, некоторые издание могут ожидать своей адаптации больше года, поскольку, во-первых, комиксов производится немало, а во-вторых, перевод каждого такого издания занимает два трудоемких этапа. Первый этап – непосредственно литературный перевод текста комикса с английского языка. Этот этап достаточно сложно автоматизировать, поскольку перевод должен отражать все литературные тонкости оригинала. С этой задачей прекрасно справляются люди, а самое важное – этот процесс для человека интересен, это настоящее творчество, и многие люди специально занимаются этим как хобби, которое приносит им удовольствие. Второй же этап приносит удовольствие человеку гораздо меньше. Этот этап происходит в графическом редакторе. При успешном завершении первого этапа переводчик должен стереть английский текст на каждой странице комикса в графическом редакторе, а затем вписать туда свой перевод, при этом ему необходимо учитывать тот факт, что текст на целевом языке может быть больше или меньше, таким образом, ему, помимо вставки текста, приходится заниматься изменением размера шрифта и междустрочных интервалов. Второй этап представляет собой монотонный труд, который затрачивает много человеко-часов. В 2017 году я участвовал в небольшом стартапе, целью которого было создать программное обеспечение, которое автоматизирует второй этап, то есть автоматически заменяет текст в тестовых облаках на перевод на целевом языке.

Постановка задачи

В рамках данной работы были поставлены следующие задачи.

- Разработка модуля для вычленения с изображения диалоговых облаков с помощью библиотек зрения
- Удаление текста внутри диалоговых облаков
- Изменение размера шрифта и междустрочного интервала в соответствии с объемами текста на целевом языке
- Вставка перевода на целевом языке
- Тестирование

Обзор используемых технологий

Используемый язык программирования – Python.

Библиотеки/фреймворки:

1) OpenCV.

   OpenCV предоставляет оптимизированную в реальном времени библиотеку, инструменты и оборудование компьютерного зрения. Он также поддерживает выполнение моделей для машинного обучения (ML) и искусственного интеллекта (AI).[1] Эта библиотека распространяется по лицензии BSD, что означает, что ее можно использовать в коммерческих и академических целях.[2]

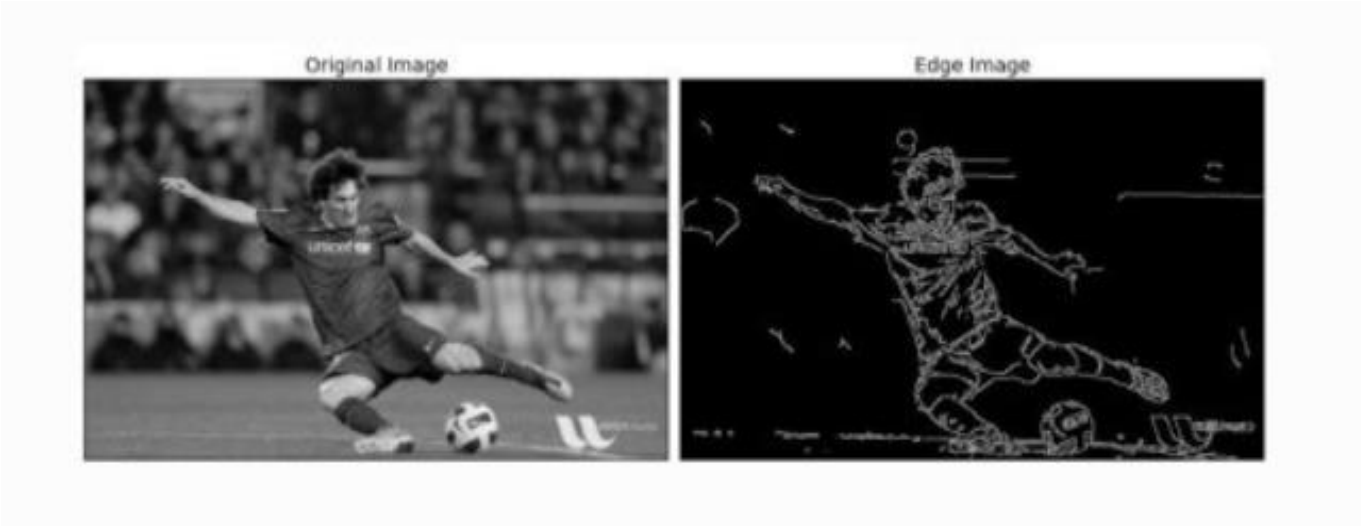
2) NumPy.

   Библиотека с открытым исходным кодом. В работе задействовалась для использования многомерных массивов.

3) Множество встроенных библиотек в Python. Например, модуль re использовался для разбиения слов на слоги и переноса частей слова на новую строчку в текстовом облаке.

Реализация

В основном цикле программы для каждой картинки комикса реализован следующий алгоритм:
Создается **cv2.Canny()** по переданному изображению.[3]
Пример работы:



Далее с помощью **cv2.findContours()** находим все контуры на изображении. Контуры – это простые кривые, точки которых расположены близко друг к другу и имеют одинаковые цвет и интенсивность. В OpenCV происходит поиск белых контуров на черном фоне.[4] Так мы пытаемся найти все текстовые облака на изображении. Очевидно, что такого способа недостаточно, так как может возникнуть большое количество других контуров, не являющихся текстовыми облаками. Поэтому после нахождения контуров мы проходим по массиву с ними и удаляем те, которые не подходят нам по некоторым признакам. По завершению этого этапа мы применяем **cv2.GaussianBlur()** к изображению и снова ищем на нем контуры. Применение блюра, а затем взятие контуров с полученного изображения позволяет четче определить границы текстовых облаков, а так же избавиться еще от нескольких типов коллизий.

После этого происходит чтение файла с переводом для текущего изображения. Для соответствующего текстового облака выбирается соответствующий перевод.

Далее мы пытаемся вписать в рассматриваемый контур прямоугольник, состоящий из строк вида: высота шрифта -> междустрочный интервал, высота шрифта и тд.. Вычисляем, сколько строк с текущими высотой шрифта и междустрочным интервалом займет текст перевода. Если перевод был выполнен на русский язык, то тогда чаще всего строк оказывается больше и приходится уменьшать высоту, ширину шрифта и междустрочный интервал.

После того, как мы выяснили, какие высоту, ширину и междустрочный интервал мы используем, создаем большую строку с переносами, которую мы и будем вписывать в контуры. Если слово не помещается на строчку, то мы должны разбить его на слоги и пытаться вписать по слогу в строчку, а когда какой-то слог не поместился, то ставим перенос строки, а после него дописываем оставшиеся слоги.

Алгоритм разбиения слова на слоги:

```
import re

def rec(s, r):
    if (r[-1].find(s)) != -1:
        r[-1] = s
    else:
        r += [s]
    result = ''
    for f in functions:
        if f(s, r) != None:
            break
    return [result, r]

def rule_1(word, r):
    s = re.findall(r'[а-я][йъь][а-я][а-я]',word)
    if s == []:
        return None
    else:
        s = s[0]
        t = re.sub(s, s[:2] + '-' + s[2:], word)
        p = t.split('-')
        return rec(p[0], r)[0] + '-' + rec(p[1], r)[0]
def rule_2(word, r):
    s = re.findall(r'[а-я][аеёиоуыэюя][аеёиоуыэюя][а-я]',word)
    if s == []:
        return None
    else:
        s = s[0]
        t = re.sub(s, s[:2] + '-' + s[2:], word)
        p = t.split('-')
        return rec(p[0], r)[0] + '-' + rec(p[1], r)[0]
```

```
def rule_3(word, r):
    s = re.findall(r'[аеёиоуыэюя][бвгджзклмнпрстфхцчшщ][бвгджзклмнпрстфхцчшщ][аеёиоуыэюя]',word)
    if s == []:
        return None
    else:
        s = s[0]
        t = re.sub(s, s[:2] + '-' + s[2:], word)
        p = t.split('-')
        return rec(p[0], r)[0] + '-' + rec(p[1], r)[0]
def rule_4(word, r):
    s = re.findall(r'[бвгджзклмнпрстфхцчшщ][аеёиоуыэюя][бвгджзклмнпрстфхцчшщ][аеёиоуыэюя]',word)
    if s == []:
        return None
    else:
        s = s[0]
        t = re.sub(s, s[:2] + '-' + s[2:], word)
        p = t.split('-')
        return rec(p[0], r)[0] + '-' + rec(p[1], r)[0]
def rule_5(word, r):
    s = re.findall(r'[аеёиоуыэюя][бвгджзклмнпрстфхцчшщ][бвгджзклмнпрстфхцчшщ][бвгджзклмнпрстфхцчшщ][аеёиоуыэюя]',word)
    if s == []:
        return None
    else:
        s = s[0]
        t = re.sub(s, s[:2] + '-' + s[2:], word)
        p = t.split('-')
        return rec(p[0], r)[0] + '-' + rec(p[1], r)[0]

def rule_6(word, r):
    s = re.findall(r'[аеёиоуыэюя][бвгджзклмнпрстфхцчшщ][бвгджзклмнпрстфхцчшщ][бвгджзклмнпрстфхцчшщ][бвгджзклмнпрстфхцчшщ][аеёиоуыэюя]',word)
    if s == []:
        return None
    else:
        s = s[0]
        t = re.sub(s, s[:3] + '-' + s[3:], word)
        p = t.split('-')
        return rec(p[0], r)[0] + '-' + rec(p[1], r)[0]

functions = [rule_1, rule_2, rule_3, rule_4, rule_5, rule_6]
```

В результате мы знаем координаты текста, который нужно вписать в облачка, высоту, ширину, междустрочный интервал и имеем саму строку с переносами, которую нам надо вывести на изображение с использованием **cv2.putText()**.

Заключение

В ходе работы были выполнены все поставленные задачи. Полученный инструмент оказался удобен на практике, с его использованием процесс перевода издания комикса сократился на несколько часов. Однако на этапе нахождения контуров все еще оставалось немалое количество коллизий, что означало, что если контуров на изображении будет найдено меньше или же будет найдено больше, то часть текста с переводом не найдет себе места на изображении или окажется не в том месте. Это оказалось достаточно существенным недостатком, поскольку при наличии даже одной коллизии при нахождении контуров получившееся изображение с текстом оказывалось непригодным и его приходилось полностью вручную переделывать. Поэтому было принято решение переписать обнаружение контуров с использованием машинного обучения.

Список литературы

[1] Wikipedia. BSD License. — 2021. — URL: https://en.wikipedia.org/wiki/BSD_licenses

[2] OpenCV. — 2021. — URL: https://opencv.org/

[3] OpenCV. Canny Edge Detection — 2021. — URL: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html

[4] OpenCV. Contours : Getting Started— 2021. — URL: https://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html