Проектна задача

“Препознавање на текст со користење на OCR”

по предметот
“Дигитално процесирање на слика”.

Изработиле:

Самоил Јаќимовски 211036

Бранко Георгиев 213077

Ментор:

Проф. д-р Ивица Димитровски

ФИНКИ, јануари 2024

Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Содржина:

1. Вовед.....3
2. Користени библиотеки 5
3. Опис на чекори 6
3.1 Easyocr 6
3.2 Инсталација на easyocr 6
3.3 Инсталација на останати потребни библиотеки 6
3.4 Импортирање на библиотеките во код 7
3.5 Креирање на фолдер од одбраните слики 7
3.6 Инстанцирање на патеките на одбрани слики 7
3.7 Вчитување на сликите 8
3.8 Reader класа од easyocr 8
3.9 Читање на информации за текстовите од слика 9
3.10 Исцртување на добиената фигура 11
4. Целосен код на програмата 12
5. Предизвици во машинското учење на OCR 13
6. Deep Learning OCR 13
7. Појава на можни грешки 14
8. Користена литература.....15

1. Вовед

OCR претставува технологија која користи алгоритми од машинско учење и овозможува автоматско препознавање на текст од слика или скенирани документи. Оваа технологија има широка примена во различни области, вклучувајќи го преобразувањето на физички документи во електронски формат, издвојување на текст од слики за пребарување, и поддршка за автоматизирани бизнис процеси.

Оваа документација е дизајнирана со цел да обезбеди детални упатства и информации за користење на OCR технологијата. Во следните делови ќе биде објаснето како да се започне со интеграција на OCR сервис во апликацијата, како и како правилно да се користи истиот за препознавање на текст.
За софтверот, секоја слика претставува сет од пиксели во различна боја, тон… Алатките за машинско учење за препознавање на текст од слика треба да се способни да ги идентификуваат групите од пиксели кои заемно формираат некоја буква. Задачата на овие модели е уште попредизвичувачка заради неколку други фактори како што се:

- Варирање на големината и формата на фонтот
- Рачно напишан текст со различни ракописи
- Заматени или слики со низок квалитет
- Повеќе текст блокови во различни делови од сликата

Сепак, технологиите на машинско учење за OCR ги решаваат овие проблеми користејќи однапред истренирани модели (или алгоритми) за скенирање на сликата и препознавање на шеми. Овие модели се тренираат со огромен број на означени податоци. Едноставно кажано, машинското учење за OCR ја скенира сликата и препознава шеми користејќи веќе истренирани модели, а потоа го пишува текстот кој го прочитал од сликата. Овој процес се случува во неколку чекори:

- Претпроцесирање на податоци - како прв чекор, повеќето OCR технологии ја претпроцесираат скенираната слика користејќи техники како променување на големината (resizing), нормализација и noise reduction за да се подобри квалитетот на влезните податоци. На пример системот може да оцрни или избрише некои точки, да го ротира документот за мали агли за да се поправат проблемите со израмнувањето, измазнување на рабовите на текстот итн…
- Локализација на текст - задачата е да се лоцираат регионите од сликата кои содржат текст. Овој процес користи техники како детекција на рабови, детекција на објекти и анализа на контура за да го одвои текстот од сликата.

- Препознавање на текст - Од кога системот ќе ги пронајде текст регионите, го декомпозира специфичниот регион од сликата за да идентификува букви и зборови. Во оваа стаза, индивидуалните карактери се нарекуваат глифи (“glyphs”) и за нивно идентификување системот ги спојува со претходно препознаени глифи или кругови, точки и ја погодува буквата по препознаената шема. Овој процес е повеќе предизвикувачки при конвертирање на ракописен текст од дигитален формат.
- Пост-обработка на податоците - препознавањето на текст може да има некакви грешки при варијација на фонтовите, квалитетот на сликата… Пост обработката се користи за подобрување на прецизноста на резултатот. Во овој чекор системот користи поправање на грешки при спелување (spell correction) и граматички правила за корегирање на текстот. На пример може да го спореди препознаениот текст со речник или да употреби статистички методи за да се провери фреквенцијата на различни зборови во текстот. Исто така може да го форматира текстот во посакуваниот излез со употреба на нормализирање, бришење на непотребни празни места итн…

2. Користени библиотеки

Овој проект е изработен во програмскиот јазик Python, но не би можеле ништо да постигнеме без користење на надворешни готови кодови, односно библиотеки.

Нашиот код започнува со вчитување (import) на следните библиотеки: OpenCV, NumPy, Imutils, EasyOCR.

Cv2 / OpenCV

Оваа библиотека е многу популарна за компјутерска визија и процесирање на слики. Вклучува функции за израмнување на слики (image smoothing), пронаоѓање на рабови (edge detection), филтрирање на слики (image filtering) и други. Ова им овозможува на програмерите да манипулираат и подобруваат сликите за разни примени и потреби.

NumPy (Numerical Python)

е библиотека за програмски јазик Python која обезбедува многу функции и операции за манипулација со низи (arrays) и матрици, како и математички функции за обработка на податоци. Оваа високо ефикасна библиотека е оптимизирана за брзина при обработка на големи количини податоци. Бидејќи сликите ги претставуваме како матрици оваа библиотека ни е многу битна и корисна во оваа програма.

Imutils

Оваа библиотека е создадена за да ја упрости работа со OpenCV. Таа обезбедува збир на употребливи функции со цел да биде полесно да се изведуваат различни задачи за обработка на слики во областа на компјутерска визија, односно ја прави OpenCV полесна за работа на корисниците, односно програмерите.

EasyOCR

е библиотека за Python која служи за препознавање на карактери од слика. Лесна и едноставна за користење, а истовремено е брза и ефикасна. Повеке и подетално за оваа библиотека ќе видиме понатаму

3. Опис на чекори

1) Easyocr

Easyocr е библиотека во Python која служи за екстракција на текст од дадена дигитална слика. Таа библиотека е општ OCR (Object Character Recognition) кој што може да чита текст скениран од документи или слики. OCR софтверот може да биде користен за конвертирање на хартиен документ или слика во достапна електронска верзија во вид на текст. Со оваа библиотека веќе се опфатени повеќе од 80 јазици.

2) Инсталација на easyocr

Најпрвин пред да се започне со кодот потребно е соодветната библиотека да се инсталира на оперативниот систем со командата: ***pip install easyocr***. Со оваа команда овозможено ни е користење на библиотеката и сите нејзини функции во околината во која работиме.

3) Инсталација на останати потребни библиотеки

Покрај easyocr потребни ќе ни бидат уште неколку додатни библиотеки како cv2 и matplotlib. OpenCV (Open Source Computer Vision Library) е open-source е библиотека во Python за компјутерска визија и машинско учење. Таа обезбедува голем број на функционалности и алатки за процесирање на слика и видео. Како и easyocr потребно е да се инсталира и оваа библиотека со командата: ***pip install opencv-python*** и со тоа ни е овозможено користење на библиотеката заедно со сите нејзини функционалности. Исто така потребна ќе ни е и библиотеката matplotpib.pyplot која претставува колекција од функции кои ни овозможуваат некаков вид на исцртување и креирање на фигури, области за цртање во фигура, исцртува линии во област на некоја фигура итн…

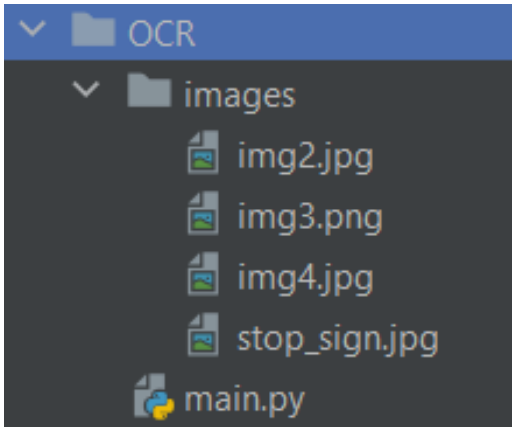
4) Импортирање на библиотеките во код

Од кога ќе ја креираме и спремиме околината за работа во која ќе биде имплементиран сервисот за препознавање на текст од слики со помош на OCR, потребно е најпрвин во кодот да ги имплементираме потребните библиотеки за да можеме да ги користиме нибните функции. Тоа го правиме на следниот начин:

```
1  import easyocr
2  import cv2
3  import matplotlib.pyplot as plt
```

5) Креирање на фолдер со избрани слики

Потребни ни се слики со кои што ќе работиме односно ќе ја тренираме и тестираме апликацијата за нејзината функционалност. Креираме пакет images во кој ќе се наоѓаат сликите кои што сме ги одбрале.



6) Инстанцирање на патеките на одбраните слики

За употреба на сликите со кои што сакаме да работиме потребно е да ги инстанцираме нивните патеки во дадени променливи кои ќе ги користиме во нашата апликација. Тоа можеме да го направиме на 2 начини односно со релативна или апсолутна патека. Во нашата

апликација искористени се релативните патеки за поточно да се специфицира локацијата на фајловите.

```
image_path1 = 'C:\\Users\\baneg\\PycharmProjects\\DPNS\\OCR\\images\\stop_sign.jpg'
image_path2 = 'C:\\Users\\baneg\\PycharmProjects\\DPNS\\OCR\\images\\img2.jpg'
image_path3 = 'C:\\Users\\baneg\\PycharmProjects\\DPNS\\OCR\\images\\img3.png'
image_path4 = 'C:\\Users\\baneg\\PycharmProjects\\DPNS\\OCR\\images\\img4.jpg'
```

7) Вчитување на сликите

Со помош на OpenCV и вградената функционалност imread() ги вчитуваме сликите од наведениот фајл. Оваа функција ја зема патеката до сликата како аргумент и враќа numpy array која што ја репрезентира сликата. Оваа листа во суштина е во вид на матрица каде секој елемент го претставува секој пиксел од сликата.

```
img1 = cv2.imread(image_path1)
img2 = cv2.imread(image_path2)
img3 = cv2.imread(image_path3)
img4 = cv2.imread(image_path4)
```

8) Reader класа од easyocr

Библиотеката easyocr нуди една класа Reader која се користи за изведување на OCR на сликите. Оваа класа претставува модел кој е добро истриниран да го лоцира и препознава текстот кој се наоѓа во даден фајл. Како прв аргумент прима листа со јазици кои моделот ги поддржува. Argumentot gpu=False означува дека библиотеката easyocr нема да се обиде да ја користи единицата за графичка обработка за да се поедностави градењето на апликацијата и да се избегнат потенцијалните проблеми со GPU. Овој модел прифаќа и други параметри кои не се искористени како што се: model_storage_directory, download_enabled, user_network_directory, recog_network, detector и

recognizer. Една од најбитните функции на овој модел е readtext() која користи OCR врз дадената слика која е проследана како прв аргумент. Од оваа функција произлегува извлечен текст од слика заедно со уште неколку информации како: bounding boxes - за анотација на регионот во кој е детектиран текстот, confidence score - индикатор за сигурноста на моделот дека детектираниот текст е точно прочитан од сликата. Како додатни параметри во оваа функција можеме да ги внесеме cls кој ја враќа класата на прочитаниот текст итн, paragraph - ако имаме дел од текст кој е поделен во повеќе линии, моделот ќе ги детектира овие линии како различен текст и ако сакаме да ги препознае како еден заеднички текст соседните линии на текст во некој вид на параграф го поставуваме овој параметар на True...

```
14
15      reader = easyocr.Reader(['ch_tra', 'en'], gpu=False)
16      text_ = reader.readtext(img1)
17
```

Оваа функција враќа листа од торки, каде секоја торка содржи информација за детектиран текст елемент. Секоја торка содржи:

- Листа од 4 координати кој го репрезентираат регионот на bounding box за детектираниот текст.
- Прочитаниот текст како стринг
- Сигурноста на моделот за точноста на прочитаниот текст

```
([[355, 177], [497, 177], [497, 241], [355, 241]], 'Hom', 0.999939709092003)
([[516, 174], [765, 174], [765, 241], [516, 241]], 'Station', 0.6706353624676109)
([[176, 248], [448, 248], [448, 348], [176, 348]], '紅磡站', 0.9991793696797584)
```

9) Читање на информации за текстовите од сликата

Следен чекор е читање на информациите за секој детектиран текст од сликата. Во еден циклус ја изминуваме секоја листа од променливата во која ги имаме прочитано информациите од сликата и најпрвин ги принтаме во конзолата. За секоја итерација променливата t која претстсавува листа од 3 елементи (објаснета во претходниот чекор) ја

делиме на 3 дела: bbox, text, score. Променливата bbox претставува листа со 4 елементи кои ги означуваат координатите во кои е детектиран текстот. Овие координати најчесто се добиени како floating-point вредности и кога ќе сакаме да го исцртаме правоаголникот за да го обележиме детектираниот текст најпрвин овие вредности ги мапираме во int вредности.
Со помош на OpenCV библиотеката и функцијата cv2.rectangle() иницијализираме правоаголник кој ги прима следните атрибути:

- img - информации за детектираниот текст
- pt1, pt2 - почетна и крајна координата за исцртување на правоаголникот во кој е детектиран текст
- rgb(x, y, z) - торка со 3 вредности во RGB формат со која се одлучува во која боја ќе биде исцртан правоаголникот за граничење на детектиран текст
- thickness - цел број со кој се нагласува со која дебелина ќе биде исцртан правоаголникот.

Потоа од OpenCV ја користиме функцијата putText() која го печати детектираниот текст како лабела околу правоаголникот за граничење и ги прима следните атрибути:

- img - информации за детектираниот текст
- text - детектираниот текст од сликата како String вредност која треба да се испечати
- pt1 - координата од која треба да се започне испишувањето на детектираниот текст
- fontFace - со помош на OpenCV и негови вградени компоненти се избира со каков фонт сакаме да биде испишан детектираниот текст
- fontScale- цел број со кој се нагласува со која големина ќе биде испишан детектираниот текст
- rgb(x, y, z) - торка со 3 вредности во RGB формат со која се одлучува во која боја сакаме да биде испишан детектираниот текст
- thickness - цел број со кој се нагласува со која дебелина ќе биде испишан детектираниот текст

```
22
23  for t in text_:
24      print(t)
25
26      bbox, text, score = t
27
28      pt1 = tuple(map(int, bbox[0]))
29      pt2 = tuple(map(int, bbox[2]))
30
31      cv2.rectangle(img, pt1, pt2, (0, 255, 0), 2)
32      cv2.putText(img, text, pt1, cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 1)
33
```

10) Исцртување на добиената фигура

Последен чекор е исцртување на добиената фигура која во неа ја содржи одбраната слика за обработка вклучувајќи ги исцртаните правоаголници кои го граничат детектираниот текст заедно со самио текст кој е прочитан од сликата. Со помош на matplotlib.pyplot и функцијата imshow() ја прикажуваме фигурата која сме ја добиле која како аргумент прима фигурата која сакаме да ја прикажеме. Како аргумент на оваа функција ќе ја предадеме сликата со помош на cv2.cvtColor() која всушност ја преобработува добиената слика во BGR формат бидејќи тој е форматот кој функцијата imshow() го поддржува. На крај само ја прикажуваме добиената фигура на екран во посебно прозорче со функцијата plt.show().

```
36
37      plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
38      plt.show()
```

4. Целосен код на програмата

```
import easyocr
import cv2
import matplotlib.pyplot as plt

image_path1 =
'C:\\Users\\baneg\\PycharmProjects\\DPNS\\OCR\\images\\stop_sign.jpg'
image_path2 = 'C:\\Users\\baneg\\PycharmProjects\\DPNS\\OCR\\images\\img2.jpg'
image_path3 = 'C:\\Users\\baneg\\PycharmProjects\\DPNS\\OCR\\images\\img3.png'
image_path4 = 'C:\\Users\\baneg\\PycharmProjects\\DPNS\\OCR\\images\\img4.jpg'

img1 = cv2.imread(image_path1)
img2 = cv2.imread(image_path2)
img3 = cv2.imread(image_path3)
img4 = cv2.imread(image_path4)

reader = easyocr.Reader(['ch_tra', 'en'], gpu=False)

img = img2

text_ = reader.readtext(img)

for t in text_:
   print(t)


   bbox, text, score = t

   pt1 = tuple(map(int, bbox[0]))
   pt2 = tuple(map(int, bbox[2]))

   cv2.rectangle(img, pt1, pt2, (0, 255, 0), 3)
   cv2.putText(img, text, pt1, cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 0, 0), 3,
cv2.LINE_AA)

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```

5. Предизвици во машинското учење на OCR

Машинското учење во OCR е софистицирана технологија, но не работи сјајно во случаи во реален свет со значајни варијации во податоците и зголемување на волуменот на податоците. Повеќето од техниките како контурна анализа, детекција на рабови, препознавање на шеми работат добро за едноставни и стандардни дигитални документи и може да се добијат точни резултати на лесен начин. Но во реалниот свет, хартиените документи може значително да се разликуваат во позицирањето и распоредот на текстот, боите, дизајнот итн… Инженерите за машинско учење се соочуваат со предизвикот на ширењето на опсегот на влезни податоци. Комплексноста се јавува од фактот што OCR оперира во интерсекција на 2 полиња:

- **Компјутерска визија (Computer Vision)** - поле за тренирање на софтвер воочување и толкување на визуелниот свет
- **Обработка на природен јазик (Natural Language Processing)** - полето за тренирање на машините да го разбираат природниот човеков јазик

Следствено, OCR моделите мора да извршат голем број на мали задачи пред постигнување на нивната главна цел.

6. Deep learning OCR

Длабокото учење е следниот чекор на развој на машинско учење. Тоа го развива OCR од компатибилен само со стандардните шаблони и системи базирани на “правила” во софистицирано AI решение кое ги анализира скенираните документи исто како човекот. Длабокото учење користи технологии наречени невронски мрежи. Тие се изградени од огромен број на внатрешни меѓусебно поврзани слоеви кои комуницираат со себе. Оваа архитектура е изградена така што секој слој обработува мал дел од проблемот и го предава на следниот слој, и е направена на овој начин со цел целосната мрежа да ги подобри точноста и способноста на моделот. Во главно Deep Learning OCR користи 2 главни типа на невронски мрежи и тоа:

- **Convoluted Neural Networks** - овие невронски мрежи содржат конволуирачки слоеви кои ги трансформираат примените податоци и ги препраќаат на следниот слој. На пример, секој слој учи да

препознае различен аспект од сликата, и со комбинирање на излезите од овие слоеви, мрежата добива поголемо разбирање на целокупната слика

- **Recurrent Neural Networks** - овие невронски мрежи работат на принципот така што користат слоеви кои имаат својства на памтење. Дозволува слоевите да памтат поранешни информации и да го користат тоа за процесирање на нов излез. Тие анализираат буква по буква, правејќи заклучоци за недостапни детали. Го препознаваат контекстот, како зависности меѓу букви и зборови, и овие мрежи можат да предвидат која е следна буква или следен збор, во зависност од претходно препознаени букви, зборови или фрази.

7. Појава на можни грешки

7.1 Грешки во распознавањето на знаци

Вакви грешки се случуваат кога една буква погрешно се чита со друг знак како на пример (“rn” може да биде грешно прочитан како “m”), може да бидат изоставени букви или цели зборови или пак додавање на дополнителни непотребни знаци/букви кои не се присутни во оригиналниот текст

7.2 Грешки во разместување

Погрешно разместување меѓу зборови или знаци, изоставање на празнини, што предизвикува зборовите да бидат конкатенирани.

7.3 Проблеми со фонтови и стилови

Тешкотии во распознавањето на знаци во украсени фонтови, неспособност за справување со варијации во големина и стил на фонтот.

7.4 Бучава и искривувања

Грешки поради шум или други искривувања на сликата, накривен/ротиран текст кој не е правилно порамнет.

7.5 Проблеми со резолуција и квалитет

Слики со ниска резолуција може да предизвикаат промашување на знаците, губењењ на детали во компримирани слики предизвикувајќи грешки.

7.6 Предизвик при распознавање на ракопис

Тешкотии при распознавање на ракописен текст поради варијации во стилот на пишување на индивидуалците, неконзистентност во дебелината на линиите и писменоста.

7.7 Грешки во распознавање на јазик и контекст

Грешки во толкувањето на зборови во специфичен контекст, тешкотии со јазици што користат комплексни знаци или писма.

8. Користена литература

- ☐ https://builtin.com/data-science/python-ocr
- ☐ https://nanonets.com/blog/ocr-with-tesseract/
- ☐ https://medium.com/@dprakash05/a-complete-guide-to-build-optical-character-recognition-ocr-in-python-5bf179d47db8
- ☐ https://courses.finki.ukim.mk/course/view.php?id=2261

- [ ] https://betterprogramming.pub/a-practical-guide-to-extract-text-from-images-ocr-in-python-d8c9c30ae74b
- [ ] https://eng-mhasan.medium.com/ocr-with-deep-learning-in-python-e443970d09e4
- [ ] https://www.affinda.com/tech-ai/machine-learning-ocr
- [ ] https://www.jaided.ai/easyocr/