FIT VUT, Počítačové Vidění
Comic Sans OCR

Miloslav Číž <xcizmi00>, Daniel Žůrek <xzurek12>

22. prosince 2016

1 Zadání a cíle

Měli jsme implementovat OCR systém pomocí knihovny OpenCV, přičemž jsme mohli předpokládat dobrou kvalitu skenovaného textu (málo šumu, dobré zarovnání, bez geometrickcých deformací apod.). Mohli jsme se rovněž omezit na jeden font. Zadání jsme si upřesnili takto:

- OCR systém pomocí C++ a OpenCV, nabízející několik klasifikátorů
- Rozpoznávány budou jen znaky anglické abecedy a některé speciální znaky:
  - malá písmena: "abcdefghijklmnopqrstuvwxyz"
  - velká písmena: "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  - číslice: "0123456789"
  - speciální znaky: ".,-?"
- Rozpoznáván bude jen font Comic Sans.

Cílem bylo vyzkoušet a porovnat různé metody používané pro přepis textu. Cílem nebylo udělat obecný, velmi robustní nebo optimalizovaný OCR systém.

2 Řešení

Systém se dá rozdělit na dvě části: segmentace a klasifikace. Museli jsme také vytvořit dataset pro trénování klasifikátorů a pro testování.

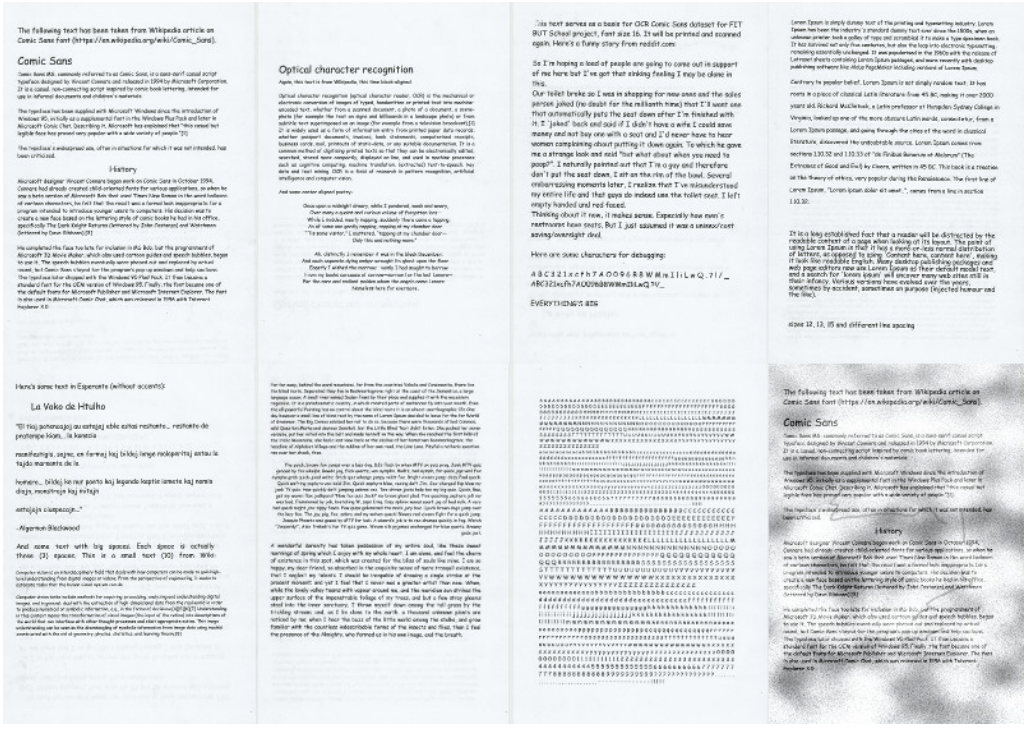
Práce v týmu byla rozdělena takto:

- Miloslav Číž - dataset, segmentace, MLP, rastr. klas., ukládání do souborů, testování, dokumentace
- David Žůrek - KNN, SVM

2.1 Dataset

Pro účely testování jsme napsali sedm A4 stran textu v programu LibreOffice Writer fontem Comic Sans. Volili jsme různé velikosti fontu (nejčastěji 12 bodů), řádkování a různá zarovnání. Texty byly většinou zkopírovány z anglických internetových stránek jako Wikipedia nebo Reddit. Stránky jsme následně vytiskli a naskenovali v rozlišení 2480×3507 pixelů. Dále jsme vytvořili zmenšené a zašuměné verze stránek. Rovněž jsme z LibreOffice dokumentů extrahovali čistý text pro pozdější testování kvality přepisu.

Dataset jednotlivých znaků jsme po implementaci segmentace vytvořili segmentací naskenovaných stránek a ruční anotací získaných výřezů. Získali jsme cca 3000 anotovaných vzorků. Dataset je dostupný spolu s kódem a skenovanými stránkami na serveru GitHub. [1] Pomocí Python skriptu jsme vytvořili průměrný obraz každého vzorku pro případné potřeby klasifikátorů.



Obrázek 1: strany 1 - 7 (zleva doprava, shora dolů) a zašuměná strana 1

[1] https://github.com/drummyfish/Comic-Sans-OCR

Obrázek 2: průměrné obrazy vzorků

2.2 Segmentace

Segmentace hledá nejprve řádky a poté jednotlivé znaky v těchto řádcích – algoritmus je pokaždé stejný, jen se segmentuje podél různých dimenzí. Toto provádí funkce `segmentation_1d`, která iteruje po řádcích/sloupcích obrazu a podle prahu průměrné intenzity vytváří spojité segmenty, které navíc nesmí být příliš malé ani velké (malé segmenty se zahazují, velké rozdělují).

Každý nalezený výřez je ještě před klasifikací upraven tak, že se posunou jeho okraje na pokud možno přesnou hranici znaku (pomocí funkce `correct_character_cutout`).

Mezery se určují podle vzdálenosti segmentů (0,3 výšky řádku).

Obrázek 3: ukázky segmentace

2.3 Klasifikace

Program nabízí následující klasifikátory:

- rastrový klasifikátor [2] - Velmi jednoduchý klasifikátor, který porovnává získaný výřez s průměrným obrazem každého vzorku pomocí korelace. Bere částečně v potaz i poměr rozměrů vzorku a průměrnou intenzitu.

- KNN, HoG příznaky - K-nearest-neighbours algoritmus na základě histogramů orientovaných gradientů.
- SVM, HoG příznaky - Support vector machine algoritmus na základě histogramů orientovaných gradientů.
- MLP - Multilayer perceptron dopředná neuronová síť. Do sítě vstupuje samotný výřez převedený do stupňů šedi a zmenšený na velikost 10×10 pixelů. Parametry sítě jsme nalezli experimentálně:
  - vrstvy: 100, 35, 20, 1
  - backpropagation trénování, maximum 50 iterací, $\epsilon = 10^{-9}$
- složený klasifikátor - Využívá všechny výše zmíněné klasifikátory ke hlasování. Není-li při hlasování většina, bere se názor nejúspěšnějšího klasifikátoru (SVM).

Natrénované modely klasifikátorů se ukládají do XML souborů, aby trénování neprobíhalo při každém spuštění znovu.

3 Testování

Pro testování jsme využili vlastní skenované stránky (viz obr. 1). Metrikou při porovnání s referenčními přepisy byla Levenshteinova vzdálenost, tzn. počet editačních kroků. [1] Využili jsme vlastní automatizované testy napsané v jazycích Python a BASH.

Dále jsme porovnávali naše řešení s kvalitním open-source OCR enginem Tesseract. [3] Tesseract je velmi komplexní a je postaven na spoustě pokročilých technologií (baseline fitting, slovníky, neuronové sítě, dvouprůchodové algoritmy apod.). [4]

Testovali jsme rovněž samotnou segmentaci tak, že jsme porovnávali výstupní a referenční texty, v nichž jsme nahradili všechny znaky kromě mezer jedním stejným znakem, takže se ignorovala klasifikace.

4 Vyhodnocení

Z tabulky 1 vidíme, že nejúspěšnější klasifikátor byl SVM. Jde-li o rychlost, zdaleka nejlepší byl rastrový klasifikátor. Tesseract měl jednoznačně nejmenší

chybu na běžných testech, při dobré rychlosti, avšak kvůli slovníkovým metodám selhával na str. 5 (Esperanto bez diakritiky), 6 (místní názvy a nestandardní slova) a 7 (seznam znaků). Náš program měl problém se segmentací textu s malým řádkováním na str. 4.

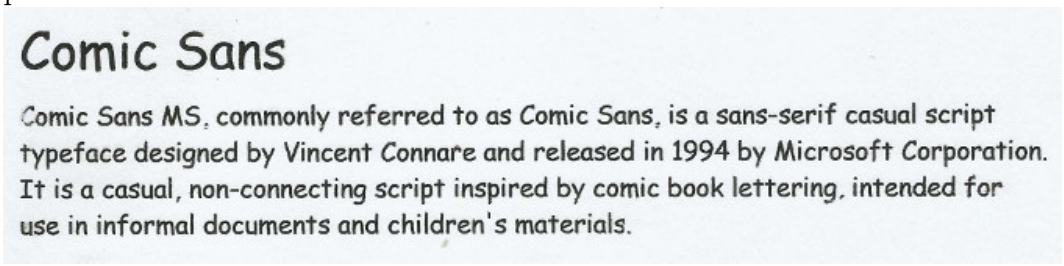
prog.	klas.	str 1	str 2	str 3	str 4	str 5	str 6	str 7	prům.
POV	segm.	130, 0.1	162, 0.1	47, 0.1	720, 0.1	163, 0.1	333, 0.2	148, 0.1	243, 0.1
POV	rast.	492, 0.4	525, 0.3	231, 0.3	969, 0.3	457, 0.3	1031, 0.5	396, 0.4	585, 0.4
POV	KNN	377, 36	445, 32	163, 29	896, 30	412, 29	878, 54	247, 42	488, 36
POV	SVM	295, 26	378, 24	114, 22	859, 23	356, 23	757, 30	176, 27	419, 25
POV	MLP	1504, 3	1381, 3	917, 3	1545, 3	1108, 3	2405, 3	1776, 3	1519, 3
POV	komb	309, 46	378, 43	124, 36	865, 36	359, 49	753, 61	186, 53	424, 45
Tess	segm.	68, 10	167, 13	64, 3	67, 8	387, 13	279, 32	2383, 7	487, 12

Tabulka 1: Výsledky testů na zmenšených skenech ve formátu [chyba (Levenshteinova vzdálenost), čas v s]. Časy zahrnují dobu trénování. Průměrný počet znaků na stranu je 2285.

program, klasifikátor	sken	čas (s)	chyba
Tesseract	page 1 (orig. rezl.)	4	26
POV, SVM	page 1 (orig. rezl.)	27	177
Tesseract	page 2 (orig. rezl.)	4	164
POV, SVM	page 2 (orig. rezl.)	26	195
Tesseract	page 1 small noise	2	2003
POV, SVM	page 1 small noise	19	2123
Tesseract	page 1 noise	3	2123
POV,SVM	page 1 noise	19	2123

Tabulka 2: dodatečné testy

Pro představu čitelnosti prezentujeme ukázku přepisu naším programem pomocí SVM klasifikátoru:




```
Comic Sans
Comic Sans MS commonly referred to as Comic Sans- is a sans serif casual scri,
```

```
lptace designed by Vincem -nnar and released in 1994 by Microsoft CorpoHtion
It is a c-ual- noTconmcti- script inspirl by comic book lJteriM imended for
use in informal documems and children-s mterials
```

5 Závěr

Implementovali jsme OCR systém pro font Comic Sans a vyhodnotili jeho aspekty. Ty jsme porovnali s používaným systémem Tesseract. Mezi hlavní zjištěné závěry patří:

- Při dobré kvalitě skenů je jednoduchý přístup velmi použitelný i na obecný text, texty jsou více či méně čitelné. Tesseract je stále mnohem kvalitnější, robustnější a obecnější, avšak někdy i pomalejší a horší při použití nestandardních jazyků.
- Spíše než dále zlepšovat klasifikátory by pravděpodobně bylo lepší zavést slovníky.
- Zmenšujeme-li výřez znaku na minimální možnou velikost, máme problém s rozlišování znaků jako ",.'".
- Bylo by lepší použít konvoluční neuronovou síť než MLP.
- Náš způsob segmentace selhává při malém řádkování.

Reference

[1] Levenshtein distance - wikipedia.
`https://en.wikipedia.org/wiki/Levenshtein_distance`.

[2] Ocr character classifier [technology portal].
`https://abbyy.technology/en:features:ocr:classifier`.

[3] tesseract-ocr/tesseract.
`https://github.com/tesseract-ocr/tesseract`.

[4] R. Smith. An overview of the tesseract ocr engine. In *Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR)*, pages 629–633, 2007.