JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 7 (2018) 191-194
Wysłane: 2017-12-10
Przyjęte: 2017-12-18

Analiza porównawcza wybranych programów do optycznego rozpoznawania tekstu

Edyta Łukasik[a,*], Tomasz Zientarski[a]

[a] Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem artykułu jest porównanie trzech programów do optycznego rozpoznawania teksu. Zdefiniowany został problem optycznego rozpoznawania tekstu i przedstawione główne jego zastosowania. Opisano działanie tej technologii i krótko scharakteryzowano najważniejsze dostępne na rynku programy realizujące omawiane zagadnienie. Następnie poddano testom wybrane programy wykorzystując dwie próbki pisma maszynowego w języku polskim. Określono szybkość procesu rozpoznawania tekstu. Poprawność rozpoznania znaków i wyrazów w analizowanym tekście została także określona.

Słowa kluczowe: rozpoznawanie tekstu, OCR; Tesseract; Ocrad; GOCR

*Autor do korespondencji.

Adres e-mail: e.lukasik@pollub.pl

Comparative analysis of selected programs for optical text recognition

Edyta Łukasik[a,*], Tomasz Zientarski[a]

[a] Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The aim of the article is to compare three programs for the optical text recognition. The problem of the optical text recognition has been defined. Next, briefly the functionality of this technology was described. The most important programs realizing the discussed problem were also characterized. The selected programs were tested using two samples of machine writing in Polish. The speed of the text recognition process was determined. The correctness of characters and words recognition in the analyzed text was also specified.

Keywords: Optical Character Recognition; OCR; Tesseract; Ocrad; GOCR

*Corresponding author.

E-mail address: e.lukasik@pollub.pl

1. Wstęp

Optyczne rozpoznawanie znaków (ang. Optical Character Recognition – OCR) jest technologią, która umożliwia konwertowanie różnego typu dokumentów do zbiorów edytowalnych i umożliwiających ich przeszukiwanie. Obejmuje ona przetwarzanie papierowych dokumentów, które są zeskanowane, plików w formacie pdf, zdjęć a także zapis komputerowy pisma odręcznego. Oprogramowanie OCR potrafi rozpoznać litery na obrazie, zbudować z nich słowa, a także zdania. Zastosowane w tej technologii algorytmy powinny gwarantować inteligencję programu, która to możliwie jak najbardziej będzie zbliżona do ludzkiego zmysłu wzroku.

Zastosowanie tej technologii pozwala na zaoszczędzenie czasu, który trzeba byłoby poświęcić na przepisanie analizowanego tekstu. Przydaje się ona w zastąpieniu pracy człowieka przy przepisywaniu dużej ilości tekstu utrwalonego na papierze, przy przenoszeniu zbiorów bibliotek do zasobów cyfrowych, a także do pomocy niepełnosprawnym. Kolejnym zastosowaniem tej technologii jest wdrożenie systemu obiegu dokumentów ze zintegrowanym modułem OCR w procesach biznesowych. Pozwala to znacznie przyspieszyć i zoptymalizować działanie firmy. Przykładem może być tutaj aplikacja służąca do przenoszenia zawartości faktur formularza elektronicznego [1].

2. Proces OCR

Klasyczny algorytm służący do optycznego rozpoznawania znaków składa się z czterech kroków [2]:

- przetwarzanie wstępne;
- analiza rozkładu;
- rozpoznawanie znaków;
- wyjście.

Pierwszy z nich obejmuje przetwarzanie wstępne (ang. preprocessing) czyli jest to odpowiednie przygotowanie obrazka do efektywniejszego działania algorytmu. Obejmuje ono binaryzację obrazka, usunięcie szumu i jego obrócenie w taki sposób, aby wyrównać w poziomie tekst. Czynności te mogą zostać zaimplementowane w programie lub wykonane przed skanowaniem. Drugim krokiem jest analiza rozkładu

tekstu (ang. layout analysis) umożliwiająca identyfikacje kolumn, tabel czy akapitów w tekście. Kolejnym krokiem procesu jest rozpoznawanie znaków (ang. character recognition) w każdej linii tekstu. Najpierw dzielona jest ona na wyrazy, a następnie w każdym słowie rozpoznawane są pojedyncze znaki. W tym miejscu algorytm korzysta z bazy danej zawierającej znaki i wybiera najlepszą wartość. Dla podniesienia skuteczności stosuje się także słowniki słów i funkcje kary dla znaków, które powodują, iż słowo nie znajduje się w słowniku [3].

Ważną sprawą jest czas trwania takiego procesu OCR i poziom błędów otrzymany na wyjściu. Jeżeli obraz jest dokładny, nie posiada szumu, a znaki są oddzielone od siebie, wtedy skuteczność algorytmu jest wysoka. Dokładność metody zależy również od rozdzielczości obrazu.

Oprogramowanie OCR z mniejszym błędem rozpoznaje tekst pisany na maszynie niż ręcznie. Powodem jest fakt, iż w pisaniu maszynowym ten sam znak zawsze wygląda tak samo, przy użyciu tej samej czcionki. Dodatkowo pismo ręczne jest o wiele bardziej zróżnicowane niż maszynowe, a skutkuje to tym, iż o wiele trudniej jest dopasować pojedyncze znaki do tych zgromadzonych w bazie danych.

Dokładność otrzymanych wyników może być mierzona na dwa sposoby:

1. na poziomie znaków (ang. character level occuracy) według wzoru

$$A_C = \frac{100 \cdot Ec}{C},$$

gdzie: *Ec* jest to liczbą błędnie rozpoznanych znaków, natomiast *C* liczba wszystkich znaków;

2. na poziomie słów (ang. word level occuracy) według wzoru

$$A_W = \frac{100 \cdot Ew}{W},$$

gdzie *Ew* jest to liczbą błędnie rozpoznanych słów, natomiast *W* liczbą wszystkich słów.

3. Biblioteki realizujące OCR – przegląd

3.1. Oprogramowanie ABBYY

Program ABBYY FineReader jest jednym z programów realizujących technologię OCR. Jego działanie opiera się o trzy podstawowe zasady: integralność, celowość i przystosowalność (ang. integrity, purposefulness, adaptability-IPA). Według zasady integralności obserwowany obiekt musi być zawsze traktowany jako „całość" składająca się z wielu powiązanych części. Zasada celowości mówi o tym, że każda interpretacja danych musi służyć jakiemuś celowi. Zasada przystosowalności oznacza, że program musi być zdolny do samodzielnej nauki. Działanie programu polega na podziale strony dokumentu na bloki tekstu, tabele, obrazy itp. Następnie wyodrębnione linie są dzielone na pojedyncze znaki, które są porównywane ze zbiorem wzorcowym. W trakcie działania algorytmu stawiane są różne hipotezy i analizowane różne warianty. Po przetworzeniu odpowiedniej liczby hipotez podejmowana jest decyzja o kształcie tekstu [4].

Program ten zawiera słowniki w 48 językach. Wynikiem działania omawianego programu jest dokument, który może być zapisany w jednym z formatów: DOC, RTF, XLS, PDF, HTML, TXT itd. lub wyeksportowany bezpośrednio do aplikacji biurowej np. Microsoft Word. Jego interfejs jest bardzo intuicyjny w obsłudze.

3.2. Silnik Tesseract

Tesseract to silnik OCR z obsługą Unicode i możliwością rozpoznawania ponad 100 języków. Moduł Tesseract-polish służy opracowaniu metody do rozpoznawania tekstów języku polskim dla programu Tesseract OCR [5]. Jest on zaopatrzony w silnik stworzony przez Google. Nie posiada on GUI.

3.3. Komponent Aspose.OCR

Aspose.OCR jest komponentem optycznego rozpoznawania tekstu, który umożliwia programistom .NET dodanie funkcjonalności OCR i OMR do tworzonych przez nich aplikacji w technologii .NET i Java. Narzędzie to dostarcza klas pozwalających programistom na optyczne rozpoznawanie tekstu i optycznych znaków z obrazów. Komponent ten jest zaimplementowany przy użyciu Managed C# i może być używany z dowolnym językiem platformy .NET [6].

3.4. Program GOCR

GOCR to program OCR opracowany na licencji publicznej GNU. Umożliwia on konwertowanie zeskanowanych obrazów tekstu na pliki tekstowe. Obecnie jest to jOCR umieszczony w sourceforge. GOCR może być używany z różnymi front-endami, co sprawia, że bardzo łatwo można go podłączyć do różnych systemów operacyjnych i architektur. Może otwierać wiele różnych formatów obrazu, a jego jakość poprawia się z dnia na dzień [7].

3.5. Aplikacja Ocrad

Ocrad jest to program OCR dostępny na licencji GNU oparty na metodzie ekstrakcji cech. Przetwarza on obrazy w formatach pbm (bitmapa), pgm (skala szarości) lub ppm (kolor) i tworzy tekst, który zapisywany jest w formatach bajtowych (8-bitowych) lub UTF-8. Zawiera on także również analizator układu tekstu, który może oddzielać kolumny lub bloki tekstu zwykle znajdowane na drukowanych stronach. Może on być używany jako samodzielna aplikacja konsolowa lub jako backend do innych programów [8].

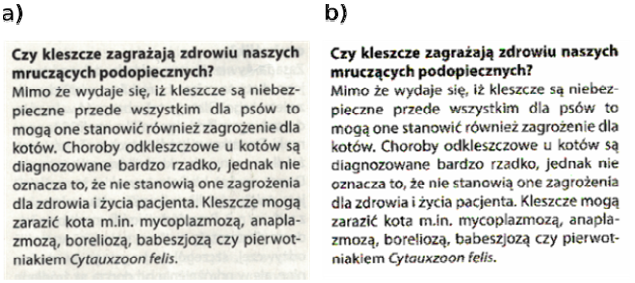
4. Cel pracy

Celem niniejszej pracy jest wskazanie aplikacji, która w zeskanowanych danych tekstowych rozpozna jak najwięcej poprawnego tekstu. Określony zostanie także wpływ palety kolorów w zeskanowanych obrazach wejściowych na poprawność i szybkość procesu rozpoznawania tekstu.

5. Analiza efektywności rozpoznawania tekstu przez wybrane programy OCR

W niniejszej pracy porównane zostały trzy aplikacje służące do optycznego rozpoznawania tekstu: Tesseract, GOCR, Ocrad.

Jako dane testowe użyto zeskanowanego z rozdzielczością 300dpi fragmentu tekstu. Wykonano dwie próbki, których fragmenty pokazano na Rysunku 1. Pierwsza próbka (T1) zawierała zeskanowany w pełnej palecie kolorów tekst, druga (T2) powstała z pierwszej przez redukcję palety kolorów do odcieni szarości. Tak otrzymane próbki zostały poddane procesowi OCR. W trakcie testów mierzono czas trwania procesu OCR. Ponadto obliczano dwie wielkości charakteryzujące jakość otrzymanego tekstu. Procentowy błąd poprawności rozpoznania wyrazów (A_W) oraz znaków (A_C) w stosunku do oryginału.



Rys. 1. Zeskanowany tekst do pliku w przestrzeni kolorów RGB (a) oraz w odcieniach szarości (b)

Wymienione we wstępie programy zostały zainstalowane ze standardowymi parametrami na komputerze klasy PC (i5M560, 8MB RAM, linux 64bit).

W pierwszej kolejności mierzono czas potrzebny do rozpoznania tekstu w analizowanym obrazie, przy czym czas mierzono od momentu uruchomienia skryptu do jego zakończenia. Wyniki przedstawiono w Tabeli 1.

Tabela 1. Czas trwania procesu rozpoznawania tekstu [s]

Próbka	Tesseract ver. 3.03	Ocrad ver. 0.22	GOCR ver 0.49
T1	9,11	0,137	3,11
T2	8,02	0,10	2,15

Czas potrzebny do rozpoznania tekstu w zeskanowanym obrazie nie był zbyt długi. Widać jednak wyraźnie dwie zależności. W każdym przypadku czas potrzebny do analizy pliku ze zubożoną paletą kolorów był zawsze nieznacznie krótszy od tego z pełną paletą. Ponadto, najkrótszy czas osiągnął program o nazwie Ocrad, ponad 60 razy szybciej od najwolniejszego programu.

Po testach szybkości przyszedł czas na testy jakościowe. Najprostszą i zarazem najczytelniejsza metodą oceny jest policzenie liczby błędnych wyrazów i znaków w stosunku do wzorcowego tekstu. Przeprowadzono obliczenia błędu poprawnego rozpoznania wyrazów i znaków w wynikowym tekście. Otrzymane wyniki przedstawiono w Tabeli 2.

Tabela 2. Błąd poprawnego rozpoznania wyrazów/znaków w analizowanym tekście

Nazwa aplikacji	Próbka T1		Próbka T2	
	Aw	Ac	Aw	Ac
Tesseract	6,78%	1,32%	6,76%	1,38%
Ocrad	46,43%	7,38%	48,23%	7,67%
GOCR	13,11%	2,61%	12,56%	2,11%

Testowe próbki zawierały 244 wyrazy składające się z 1896 znaków. Najprecyzyjniejszym programem okazał się produkt o nazwie Tesseract dający plik wyjściowy z najmniejszą liczbą niepoprawnych rozpoznań, zarówno na poziomie wyrazów jak i znaków. Najgorszy wynik osiągnął Ocrad.

6. Wnioski

Przeprowadzone badania pokazały, że niewątpliwym liderem wśród programów do OCR jest Tesseract. Stworzony w latach 90 dwudziestego wieku przez firmę HP, a obecnie udostępniany przez Google [5]. Nie jest on zbyt szybki, ale daje bardzo dobre jakościowo pliki wynikowe.

Ponadto, testowane programy w obecnych wersjach nie są wrażliwe na użytą paletę kolorów pliku graficznego zawierającego analizowany obraz. Dotyczy to szczególnie programu Tesseract, który w wersji 2.0 dawał procent niepoprawnych rozpoznań na poziomie 70-90% gdy obraz był zeskanowany z pełna paletą barw [9]. W obecnej wersji programu zostało to poprawione, jak pokazują otrzymane wyniki.

Zastanawiające rezultaty otrzymano dla programu Ocrad. Rozpoznawanie tekstu przebiegało błyskawicznie, ale otrzymane wyniki nie nadawały się do dalszej obróbki. W pracy [9] pokazano, że daje on wyniki lepsze od programu GOCR. Z kolei autor w tekście [10] stwierdza, że Ocrad daje dobrą rozpoznawalność tekstu ale tylko, gdy w oryginalnym skanowanym dokumencie była użyta standardowa czcionka, czyli Times New Roman. Okazuje się, że program ten stosuje zupełnie inne algorytmy rozpoznawania liter w porównaniu do pozostałych omawianych. Oparte są one na binarnym porównywaniu kształtu znaków [8]. Daje mu to bardzo dużą szybkość działania i bardzo dużą wrażliwość na kształt użytej czcionki. W rozważanym przypadku na niską dokładność wyniku wpłynął fakt analizy tekstu z polskimi znakami diakrytycznymi.

Literatura

[1] Bieniecki, Analiza wymagań dla metod przetwarzania wstępnego obrazów w automatycznym rozpoznawaniu tekstu,

http://wbieniec.kis.p.lodz.pl/research/files/05_Bronislawow_OCR.pdf [12.11.2017].

[2] Tobias Blanke, Michael Bryant, Mark Hedges, Open source optical character recognition for historical research, Journal of Documentation 68 (2012), 659-683.

[3] Inad Aljarrah, Osama Al-Khaleel, Khaldoon Mhaidat, Mu'ath Alrefai, Abdullah Alzu'bi, Mohammad Rabab'ah, Automated System for Arabic Optical Character Recognition with Lookup Dictionary, Journal of Emerging Technologies in Web Intelligence 4 (2012), 362-370.

[4] Abbyy Technology Portal, https://abbyy.technology/en:start, [22.11.2017].

[5] The Tesseract open source OCR engine, http://code.google.com/p/tesseract-ocr [20.11.2017].

[6] https://products.aspose.com/ocr, [01.11.2017].

[7] GOCR open-source character recognition, http://jocr.sourceforge.net, [25.11.2017].

[8] www.gnu.org/software/ocrad/manual/ocrad_manual.html, [10.12.2017].

[9] Review of Linux OCR software, https://www.mathstat.dal.ca/~selinger/ocr-test [01.12.2017].

[10] Linux OCR Software Comparison, httpswww.splitbrain.org/blog/2010-06/15-linux_ocr_software_comparison, [02.12.2017].