SEGMENTACIJA I KLASIFIKACIJA ELEMENATA NA STRANICI DOKUMENTA

Sažetak. U ovom radu je implementiran i evaluiran sistem za segmentaciju i klasifikaciju elemenata stranice dokumenta. Dobijeni rezultati pokazuju da je bilo dovoljno jednostavno enkodiranje elemenata u vektor koji sadrži odnose globalnih atributa elementa a potom dati enkodirani vektor proslediti kao ulaz random decision forest klasifikatoru da bi se dobila tačnost od 72%. Takođe je prikazan i algoritam koji vrši hijerarhijsko segmentiranje stranice u stablo.

1. Uvod

Optičko prepoznavanje karaktera (eng. OCR - Optical Character Recognition) igra važnu ulogu u današnje vreme kao način pretvaranja fizičkih dokumenata u digitalne [1]. Međutim, pre samog prepoznavanja karaktera, dokumenti imaju specifičnu strukturu u smislu organizacije elemenata na stranici. Na primer, naučni radovi su često organizovani tako da svaka stranica ima dve kolone gde na prvoj postoji i naslov koji se širi preko cele stranice po širini. Zbog ovoga je potrebno uraditi korake pre samog prepoznavanja teksta kako bi se elementi izdvojili na stranici. Na stranicama, pored teksta, se često mogu pronaći i drugi elementi poput tabela, slika, itd. Sam tekst može biti napisan i različitim stilom i samim tim nosi različito semantičko značenje. Primer ovoga je naslov sekcije kojeg je potrebno posmatrati drugačije od paragrafa unutar sekcije jer sam naslov predstavlja neku semantičku celinu. Kao posledica ovakvog organizovanja dokumenata se javlja potreba za segmentacijom stranice, odnosno izdvajanje i kreiranje hijerarhijske strukture stranice kao i klasifikacija svakog segmentiranog dela u određene klase.

U ovom radu je implementirana segmentacija elemenata stranice tako da se dobije hijerarhijska struktura a potom je svaki element klasifikovan u predefinisane kategorije kao što su: slika, naslov, paragraf, listing, tabela itd.

U sekciji 2 je dat pregled korišćenih metoda i to izdvojenih u podsekcije gde se u sekcijama 2.1 i 2.2 opisuje ulaz i izlaz samog sistema i uvode strukture koje su korišćene. U sekciji 2.3 je opisana metoda segmentacije dok u sekciji 2.4 su opisane metode klasifikacije elemenata. U sekciji 2.5 su dati implementacioni detalji u smislu korišćenih tehnologija.

U sekciji 3 je dat regled rezultata opisanih metoda kao i diskusija rezultata u smislu njihovog praktičnog značaja.

U sekciji 4 je dat zaključak kao i predlozi za buduće radove.

Sekcija 5 daje pregled literature koja je korišćena.

2. Metode

2.1. **Ulazni podaci.** Dokument je lista stranica gde je svaka stranica P predstavljena kao slika odnosno matrica piksela. Ovo je moguće uraditi procesom skenirana ručno napisanog dokumenta, ili procesom renderovanja PDF dokumenata uz pomoć postojećih alata kao što je Poppler [2].

Dakle dokument se može posmatrati kao sekvenca slika njegovih stranica

$$D = (P_1, P_2, ..., P_n)$$

gde je

$$0 \leq P_i(x, y, c) \leq 1$$

vrednost piksela na i-toj stranici na poziciji (x,y) na kanalu c (slika je u RBG formatu boja odnosno c je element skupa {R, G, B}).

Segmentacija i klasifikacija se radi na svakoj stranici izolovano, odnosno rezultati segmentacije i klasifikacije na i-toj stranici su nezavisni od rezultata na j-toj stranici.

2.2. **Izlazni podaci.** Izlaz sistema je skup elemenata gde je svaki element uređena torka (x,y,w,h,k) gde:

- x predstavlja x-koordinatu gornjeg levog piksela datog elementa
- y predstavlja y-koordinatu gornjeg levog piksela datog elementa
- w predstavlja širinu elementa na stranici
- j predstavlja visinu dokumenta na stranici
- k je klasa elementa dobijena od strane klasifikatora

Pored ovog skupa, izlaz je i funkcija

$$p : A \to A$$

koja određuje hijerarhijski raspored elemenata kao stabla gde za element x važi da je deo elementa p(x).

Klasa dokumenta je klasa iz predefinisanog skupa mogućih klasa i to

$$k \in \{\text{NONE}, \text{IMAGE}, \text{TEXT}, \text{FORMULA}, \text{HEADING}, \text{LISTING}, \text{TABLE}\}$$

2.3. **Segmentacija.** Za segmentaciju je korišćen xy-cuts algoritam [3]. Algoritam radi hijerarhijsko segmentiranje dokumenta na sledeći način. Neka

$$e = (x, y, w, h)$$

predstavlja određeni deo dokumenta. Algoritam za dati element izbacuje elemente

$$e_1, ..., e_l$$

za koje važi

$$p(e_i) = e, 1 \leq i \leq l$$

. U slučaju semgentiranja stranice, kreće se od same stranice kao korenskog elementa stabla i stranica se rekurzivno deli na manje delove po sledećem postupku:

```
Algorithm xy-cuts(P: stranica, E = (x,y,h,w), metod)
    neka je grayscale varijanta P(x,y) = (P(x,y,R) + P(x,y,G), P(x,y,B)) / 3
    B = P
    for x from E.h to E.y + E.h do
for y from E.y to E.x + E.w
    B[x,y] = 1 if E[x,y] > 0.5 else 0
    metod je iz skupa {VERTIKALNO, HORIZONTALNO}
    suma[i] = 0 for i s.t. 0 <= i <= max(P.h, P.w)
    start = 0
    if metod = VERTIKALNO
start = E.y
for y from E.y to E.y+E.h do
    suma[y] = 0
```

```
    for x from E.x to E.x + E.w do
suma[y] += P[x,y]
    if suma[y] > 0 then
suma[y] = 1
    else
start = E.x
for x from E.x to E.x+E.w do
    suma[x] = 0
    for y from E.y to E.y + E.h do
suma[x] += P[x,y]
    if suma[x] > 0 then
suma[x] = 1
    state = 1
    c = 0
    whitespace = 0
    result = []
    if suma.length = 0 then return []
    for i from start to suma.length do
p = suma[i]
if p == 0 and state == 1: c += 1
elif p == 1 and state == 0:
    c += 1
    whitespace = 0
elif p == 0 and state == 0:
    if whitespace >= max_whitespace:
result.add(i-c)
result.add(i)
c = 1
state = 1
    else:
c += 1
whitespace += 1
elif p == 1 and state == 1:
    c = 1
    state = 0
    whitespace = 0
    if suma[suma.length-1] == 0 and state == 0:
result.add(suma.length - c)
result.add(suma.length)
    sort(result)
    return result
```

Algoritam izračunava gde je potrebno element saseći kako bi se dobili manji elementi. Postoje dva načina po kojima on može da podeli element a to je vertikalno i horzontalno. Mesta gde se vrši podela jesu ona mesta gde ima mnogo belih piksela odnosno kada je suma crnih piksela u datom redu (za vertikalno presecanje) odnosno u datoj koloni (za horizontalno presecanje) jednaka nuli.

Nakon ovog algoritma, moguće je element podeliti na manje tako što se uzmu podelementi takvi da su presecanja njihove granice. Algoritam se ponavlja rekurzivno

sa obrnutim metodom, odnosno ako je metod dobijenih podelemenata bio VERTIKALNO onda se algoritam poziva rekurzivno za svaki podelement sa metodom HORIZONTALNO i obrnuto.

2.4. **Klasifikacija elemenata.** Nakon segmentacije, elementi su klasifikovani upotrebom različitih vrsta klasifikatora. Pre same klasifikacije bilo je potrebno uraditi ekstrakciju određenih karakteristika i napraviti vektor kako bi on bio pogodan za određeni klasifikator.

2.4.1. *Metode koje enkodiraju samo osnovne podatke o elementu.* Jedna od najjednostavnijih metoda je da se enkodiraju samo osnovni podaci o elemtu. U daljem delu teksta će ova metoda biti referencirana kao `simple` preprocesor:

`simple`: Element je enkodiran kao vektor [E.x, E.y, E.h, E.w], odnosno njegova pozicija i veličina su predstavljeni kao vektor.

Međutim, intuitivno se da zaključiti da zapravo odnosi ovih atributa na slici mogu predvideti klasu nekog elementa. Pa tako, na primer, paragraf često nema jednak odnos visine i širine itd. Zbog ovoga, drugi način enkodiranja, koji još uvek zadržava jednostavnost, jeste:

`img_attrs`: Element je enkodiran kao vektor: [E.h * E.w, E.w / (E.h*E.w), E.h / E.w, prosečna vrednost piksela u E]

2.4.2. *Metode enkodiranja bazirane na vrednostima piksela.* Druga dva metoda enkodiranja koja su korišćena su bazirana na vrednostima piksela samog elementa.

Pre svega, potrebno je naglasiti da za svaki element vektori moraju biti iste veličine, pa tako je potrebno uraditi svođenje slika svih elemenata na jednaku veličinu:

$$R(x, y) = (resize)(P, L, L)(E.x + x, E.y + y)$$

gde funkcija resize za datu sliku vraća sliku veličine LxL. U ovom radu je odabrana fiksna vrednost L=50.

Sada je moguće izračunati enkodirane vektora po dva modela:

`pixels`: [R(x,y) za svako (x,y) na slici R veličine LxL]

`histogram`:

$$[\sum_{x=0}^{L} R(x,0), \sum_{x=0}^{L} R(x,1), ..., \sum_{x=0}^{L} R(x,L), \sum_{y=0}^{L} R(0,y), ..., \sum_{y=0}^{L} R(L,y)]$$

2.4.3. *Klasifikatori.* U ovom radu su enkodirani vektori dati kao ulaz klasifikatoru. Klasifikatori korišćeni u ovom radu su neuronska mreža i random decision forest. Razlog za odabir ovih metoda je bio što su oni pokazivali dobre rezultate u praksi za slične probleme [4] [5]

2.5. **Implementacija.** Sistem je implementiran i testiran u programskom jeziku `Python` upotrebom `scikit-learn` biblioteke. Kako bi se izlazni podaci mogli koristiti nezavisno od implementiranog sistema, izlaz je sniman u kolekciju `JSON` objekata u datoteke na disku. `JSON` objekti imaju strukturu stabla elemenata koja je definisana u sekciji 2.2.

3. Rezultati i diskusija

Izmerena je tačnost klasifikatora za date metode enkodiranja elemenata u vektore. Skup podataka je podeljen u skup za obuku i skup za testiranje i izvršena je k-fold kros validacija.

U sledećoj tabeli su dati rezultati za klasifikatore korišćene u kombinaciji sa metodom enkodiranja elemenata u vektor:

	Metod enkodiranja	Klasifikator	Tačnost (%)
1	`histogram`	RF	71.792
2	`img_attrs`	RF	72.034
3	`img_attrs`	one rule	63.153
4	`pixels`	NN	38.451
5	`simple`	NN	42.345
6	`simple`	RF	70.422
7	`simple`	one rule	63.216

Iz datih rezultata se vidi da je najbolje rezultate dao random decision forest klasifikator za enkodiranje koje nije uključivalo vrednosti samih piksela. Ovakav rezultat se može objasniti činjenicom da random decision forest radi dobro kada su ulazni podaci interpretabilni na neki način. Kada već postoji određena intuicija oko odnosa ulaznih podataka što važi za metode enkodiranja koje ne uzimaju u obzir sirove vrednosti piksela nego enkodiraju globalne atribute datog segmenta na slici kao što su odnosi širine i dužine ili odnos širine i površine ili prosečna vrednost piksela.

Sa druge strane, može se videti da je neuronska mreža dala dosta lošije vrednosti nego referentni one rule klasifikator. Objašnjenje za ovu situaciju je izuzetno nizak broj uzoraka korišćenih za obučavanje modela. U obučavanju modela su korišćene samo 3 stranice dokumenta iz razloga što je bilo izuzetno vremenski zahtevno napraviti veći skup labeliranih podataka pa samim tim je i za očekivati lošije rezultate klasifikatora koji zahtevaju izuzetno veliki skup podataka za obuku.

4. Zaključak

Random decision forest je pokazao najbolje rezultate za izuzetno jednostavan metod enkodiranja koji ne uzima mnogo procesorskog vremena. Na 7 klasa dobijeni rezultati su zadovoljavajući jer u praksi bi bi pogrešno klasifikovani elementi bili ispravljeni ručno. Ono što nedostaje ovom radu jeste analiza klasifikatora na većem skupu podataka, kao i određivanje matrice konfuzije za date klase. Praktično gledano, pogrešna klasifikacija paragrafa u listing je manje značajna od, na primer, pogrešne klasifikacije teksta u sliku.

Literatura

[1] Shunji Mori, Hirobumi Nishida, and Hiromitsu Yamada. *Optical character recognition*. John Wiley & Sons, Inc., 1999.

[2] Poppler library (poppler.freedesktop.org - pristupljeno 24.09.2022.).

[3] Jaekyu Ha, Robert M Haralick, and Ihsin T Phillips. Recursive xy cut using bounding boxes of connected components. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 2, pages 952–955. IEEE, 1995.

[4] Semere Kiros Bitew. Logical structure extraction of electronic documents using contextual information. Master's thesis, University of Twente, 2018.

[5] Yi He. Extracting document structure of a text with visual and textual cues. Master's thesis, University of Twente, 2017.