SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2542

AUTOMATSKO PREPOZNAVANJE TISKANOG TEKSTA

Nikola Zadravec

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2542

AUTOMATSKO PREPOZNAVANJE TISKANOG TEKSTA

Nikola Zadravec

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

DIPLOMSKI ZADATAK br. 2542

Pristupnik:	**Nikola Zadravec (0036485406)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	izv. prof. dr. sc. Marko Čupić

Zadatak: **Automatsko prepoznavanje tiskanog teksta**

Opis zadatka:

Računalno prepoznavanje teksta sa slike (primjerice dobivene skeniranjem dokumenta) težak je problem koji je moguće rješavati različitim pristupima. Tekst pri tome može biti rukom pisan, ili pak tiskan. Najjednostavniji se pristupi temelje na nizu slijednih koraka poput segmentacije pojedinih slova, klasifikacije segmentiranog slova, grupiranja segmentiranih slova u riječi i slično. U okviru ovog diplomskog rada potrebno je proučiti postupke za prepoznavanje tiskanog teksta, pri čemu se treba ograničiti na slučaj kada su riječi pisane hrvatskim jezikom (eventualno engleskim). Potrebno je razviti prototipnu programsku implementaciju sustava koji će na ulazu dobiti sliku odskenirane stranice teksta (primjerice, stranicu knjige), a na izlazu dati prepoznati tekst. U okviru rada potrebno je napraviti i vrednovanje razvijenog sustava. Radu je potrebno priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 28. lipnja 2021.

SADRŽAJ

1. Uvod

Računalno prepoznavanje teksta/znakova sa slike (engl. Optical character recognition, OCR) je pretvorba slike tiskanog ili rukom pisanog teksta u strojno kodirani tekst (oblik razumljiv računalima). Slika teksta može biti skenirani dokument, fotografija dokumenta ili fotografija scene (na primjer tekst na prometnim znakovima).

OCR se često koristi za digitalizaciju tiskanih tekstova kako bi se mogli elektronički uređivati, pretraživati, kompaktnije pohranjivati, prikazivati na internetu i koristiti u računalnim procesima kao što su strojno prevođenje, pretvaranje teksta u govor, itd. OCR je područje istraživanja u prepoznavanju uzoraka (engl. pattern recognition), umjetnoj inteligenciji i računalnom vidu.

Ovaj rad će se baviti postupcima za prepoznavanje tiskanog teksta, pri čemu će se ograničiti na slučaj kada su riječi pisane hrvatskim jezikom. Razvit će se prototipna programska implementacija sustava koji će na ulazu dobiti sliku odskenirane stranice teksta (primjerice, stranicu knjige), a na izlazu dati prepoznati tekst. U okviru rada će se napraviti i vrednovanje razvijenog sustava.

2. Postupci prepoznavanja tiskanog teksta

Optičko prepoznavanje znakova (engl. Optical Character Recognition, OCR) pretvara sliku dokumenta u tekst koji se može uređivati. Primjer skenirane stranice knjige prikazan je na slici 2.1. Rezultat pretvorbe skenirane stranice u tekst online OCRom[1] možemo vidjeti na slici 2.2.

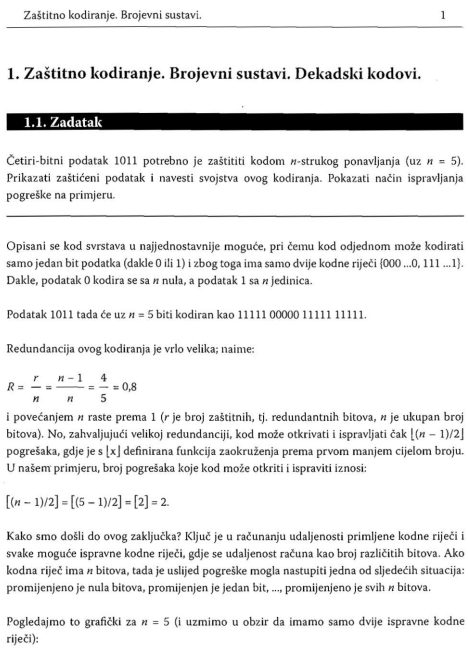
Prepoznavanje izoliranih znakova može se izvesti kao standardni problem raspoznavanja uzoraka (engl. pattern recognition), ali za cjeloviti OCR sustav koji na ulazu prima sliku dokumenta treba učiniti puno više (npr. pronalaženje i odvajanje teksta od netekstovnih područja stranice) (Shafait, 2011).

Blok dijagram tipičnog OCR sustava prikazan je na slici 2.3. Slika dokumenta nakon binarizacije i predobrade prolazi kroz modul za segmentaciju stranice i nalaze se područja teksta. Područje teksta se dijeli na redove teksta, koji se dalje dijele na riječi i znakove. Značajke se izračunavaju iz slike znaka i šalju modulu za prepoznavanje znakova. Naknadna obrada uključuje korake kao što su korekcija pogrešaka (Doermann i Tombre, 2014).

Performansa OCRa na savršeno segmentiranim slikama tiskanih znakova latinice dosegla je stopu pogrešaka manju od 1%, pa stoga ima vrlo malo prostora za poboljšanje. Segmentacija često dovodi do više OCR pogrešaka nego izolirano prepoznavanje znakova, zbog problema poput znakova koji se dodiruju te razlomljenih znakova. Većina OCR sustava usvojili su pristupe segmentacije temeljene na prepoznavanju (engl. recognition based segmentation) kako bi se smanjile pogreške u segmentaciji. Konkretno, segmentiramo znakove s ciljem što boljeg prepoznavanja segmentiranih znakova (Doermann i Tombre, 2014; Parker, 2010).

Idući odjeljak opisuje osnovne algoritme za obradu slika koji se koriste u procesu analize dokumenata na koje ćemo se referencirati u ostatku rada. Nakon toga slijede odjeljci u kojima se detaljno razmatra svaki od koraka blok diagrama sa slike 2.3.

[1] korišteni OCR: https://www.onlineocr.net/



Zaštitno kodiranje. Brojevni sustavi. 1

1. Zaštitno kodiranje. Brojevni sustavi. Dekadski kodovi.

1.1. Zadatak

Četiri-bitni podatak 1011 potrebno je zaštititi kodom n-strukog ponavljanja (uz $n = 5$). Prikazati zaštićeni podatak i navesti svojstva ovog kodiranja. Pokazati način ispravljanja pogreške na primjeru.

Opisani se kod svrstava u najjednostavnije moguće, pri čemu kod odjednom može kodirati samo jedan bit podatka (dakle 0 ili 1) i zbog toga ima samo dvije kodne riječi {000 ...0, 111 ...1}. Dakle, podatak 0 kodira se sa n nula, a podatak 1 sa n jedinica.

Podatak 1011 tada će uz $n = 5$ biti kodiran kao 11111 00000 11111 11111.

Redundancija ovog kodiranja je vrlo velika; naime:

$$R = \frac{r}{n} = \frac{n-1}{n} = \frac{4}{5} = 0{,}8$$

i povećanjem n raste prema 1 (r je broj zaštitnih, tj. redundantnih bitova, n je ukupan broj bitova). No, zahvaljujući velikoj redundanciji, kod može otkrivati i ispravljati čak $\lfloor (n-1)/2 \rfloor$ pogrešaka, gdje je s $\lfloor x \rfloor$ definirana funkcija zaokruženja prema prvom manjem cijelom broju. U našem primjeru, broj pogrešaka koje kod može otkriti i ispraviti iznosi:

$$\lfloor (n-1)/2 \rfloor = \lfloor (5-1)/2 \rfloor = \lfloor 2 \rfloor = 2.$$

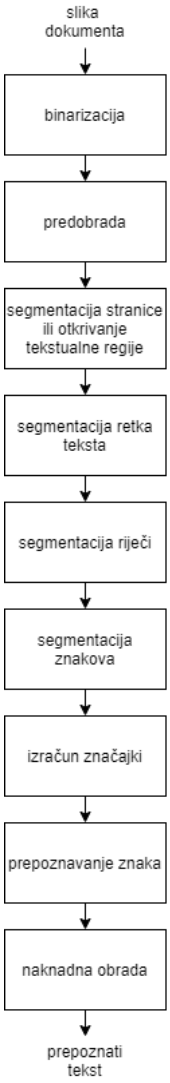
Kako smo došli do ovog zaključka? Ključ je u računanju udaljenosti primljene kodne riječi i svake moguće ispravne kodne riječi, gdje se udaljenost računa kao broj različitih bitova. Ako kodna riječ ima n bitova, tada je uslijed pogreške mogla nastupiti jedna od sljedećih situacija: promijenjeno je nula bitova, promijenjen je jedan bit, ..., promijenjeno je svih n bitova.

Pogledajmo to grafički za $n = 5$ (i uzmimo u obzir da imamo samo dvije ispravne kodne riječi):

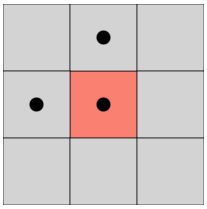
Slika 2.1: Skenirana stranica knjige

Zaštitno kodiranje. Brojevni sustavi.
1
1. Zaštitno kodiranje. Brojevni sustavi. Dekadski kodovi.
1.1. Zadatak
Četiri-bitni podatak 1011 potrebno je zaštititi kodom n-strukog ponavljanja (uz n = 5).
Prikazati zaštićeni podatak i navesti svojstva ovog kodiranja. Pokazati način ispravijanja
pogreške na primjeru.
Opisani se kod svrstava u najjednostavnije moguće, pri čemu kod odjednom može kodirati
sarno jedan bit podatka (dakle 0 ili 1) i zbog toga ima samo dvije kodne riječi {000 ...0, 111 ...1).
Dakle, podatal< 0 kodira se sa n nula, a podatak 1 sa n jedinica.
Podatak 1011 tada će uz n = 5 biti kodiran kao 11111 00000 11111 11111.
Redundancija ovog kodiranja je vrlo velika; naime:
r n - 1 4 R = - - 0,8
n n 5 i povećanjem n raste prema 1 (r je broj zaštitnih, tj. redundantnih bitova, n je ulcupan broj
bitova). No, zahvaljujući velikoj redundanciji, kod može otkrivati i ispravljati čak 1.(n — 1)/2.1
pogrešaka, gdje je s [x] definirana funkcija zaolcruženja prema prvom manjem cijelom broju.
U našem primjeru, broj pogrešaka koje lcod može otkriti i ispraviti iznosi:
[(n - 1)12] = [(5 1)/2] = [2] = 2.
Kako smo došli do ovog zaključka? Ključ je u računanju udaljenosti primljene kodne riječi i
svake moguće ispravne kodne riječi, gdje se udaljenost računa lcao broj različitih bitova.
Ako kodna riječ ima n bitova, tada je uslijed pogreške mogla nastupiti jedna od sljedećih situacija:
promijenjeno je nula bitova, promijenjen je jedan bit, promijenjeno je svih n bitova.
Pogledajmo to grafički za n = 5 (i uzmimo u obzir da imamo samo dvije ispravne kodne
riječi):

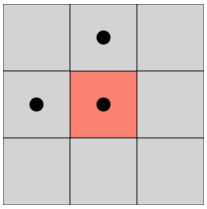
Slika 2.2: Rezultat online OCRa skenirane stranice knjige



Slika 2.3: Blok dijagram tipičnog OCR sustava



(a) 4-povezana piksela



(b) 8-povezana piksela

Slika 2.4

2.1. Algoritmi za obradu slika

Digitalna obrada slike potkategorija je digitalne obrade signala i odnosi se na upotrebu računalnih algoritama za obradu digitalnih slika. U ovom će se dijelu predstaviti osnovni algoritmi za obradu slika koji se koriste u procesu analize dokumenata (Doermann i Tombre, 2014).

Za potrebe ovog rada dovoljno je zamišljati sliku kao 2D matricu u kojoj svaki element matrice predstavlja piksel slike.

2.1.1. Označavanje povezanih komponenata

Označavanje povezanih komponenata koristi se za dodjeljivanje jedinstvene oznake svakom području slike, što omogućuje razlikovanje različitih objekata. U binarnim slikama oznaka se dodjeljuje svakom slikovnom pikselu tako da povezani pikseli imaju istu oznaku. Dva su uobičajena načina definiranja povezanih piksela za 2D sliku: 4-povezana piksela (povezana samo horizontalno ili vertikalno, tj. susjedni pikseli dijele stranicu, slika 2.4a) i 8-povezana piksela (povezana horizontalno, vertikalno ili dijagonalno, tj. susjedni pikseli dijele vrh, slika 2.4b). U obradi slike dokumenta koristi se uglavnom u fazama predobrade i segmentacije (Doermann i Tombre, 2014).

Algoritam za označavanje 4-povezana piksela[2]

- Skeniraj sliku red po red
- Kada se naiđe na crni piksel, dodijeli mu oznaku:
  - Ako je lijevi susjed bijele boje, nova oznaka dodjeljuje se trenutnom pikselu
  - Ako je lijevi susjed crne boje, njegova se oznaka kopira u trenutni piksel

[2] Algoritam je preuzet iz (Shafait, 2011).

- Ako je gornji susjedni piksel crne boje, spoji oznaku trenutnog piksela i oznaku gornjeg susjeda

Dani algoritam se lako proširi za označavanje 8-povezana piksela.

2.1.2. Algoritam zaglađivanja po smjeru trčanja (RLSA)

Na binarnim slikama bijele staze odgovaraju uzastopnim horizontalnim ili vertikalnim pozadinskim pikselima. Za svaki smjer uklanjaju se bijele staze čija je duljina manja od vrijednosti praga zaglađivanja. U procesu analize slike dokumenata, RLSA se pokazao vrlo korisnim uglavnom za predobradu, segmentaciju i analizu rasporeda elemenata na stranici dokumenta (Doermann i Tombre, 2014).

Neka su crni pikseli predstavljeni s 1, a bijela pozadina s 0. Primjenom RLSA algoritma horizontalno s pragom zaglađivanja $T = 4$ redak slike x preslikava se u y na sljedeći način:

$$x : 1100011110000010 01$$

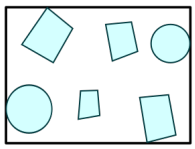
$$y : 1111111110000011 11$$

2.1.3. Houghova transformacija

Houghova transformacija podrazumjeva transformaciju iz koordinatne ravnine slike u prostor parametara, a koristi se s ciljem pronalaska linija ili krivulja koje se aproksimiraju (engl. fitting) skupinama točaka na ravnini slike. Svrha Houghove transformacije je pronalaženje primjeraka predmeta unutar određene klase oblika postupkom glasanja. Ovaj postupak glasanja provodi se u prostoru parametara, iz kojeg se kandidati za objekte dobivaju kao lokalni maksimumi u takozvanom akumulatorskom prostoru koji je konstruiran algoritmom za izračunavanje Houghove transformacije. U analizi slike dokumenta uglavnom se koristi za procjenu nagiba dokumenta, otkrivanje redaka teksta i izdvajanje značajki. Glavni nedostatak njegove primjene na velikim slikama je relativno mala brzina. Za ubrzanje Houghove transformacije može se odabrati samo nekoliko karakterističnih točaka slike ili središta povezanih komponenata kako bi se izračunao Houghov prostor (Doermann i Tombre, 2014).

2.1.4. Profili projekcije

Profil projekcije binarne slike u određenom smjeru odnosi se na zbroj crnih piksela u tom smjeru. Izračunavanjem lokalnih minimuma horizontalnih i vertikalnih projekcija



Slika 2.5: Niz geometrijskih oblika zatvorenih minimalnim ograničavajućim pravokutnikom

možemo definirati nekoliko segmenata na koje se slika može podijeliti. Profili projekcije također se koriste za otkrivanje nagiba dokumenta i izdvajanje značajki (Doermann i Tombre, 2014).

2.1.5. Minimalni ograničavajući pravokutnik

Minimalni ograničavajući pravokutnik (engl. minimum bounding rectangle, MBR), poznat i kao granični okvir (engl. bounding box, BBOX) ili omotnica, je pravokutnik minimalne površine koji obuhvaća željene objekte (u našem slučaju znakovi, riječi i retci teksta na slici) (Wikipedia contributors, 2020). Primjer graničnog okvira možemo vidjeti na slici 2.5 [3].

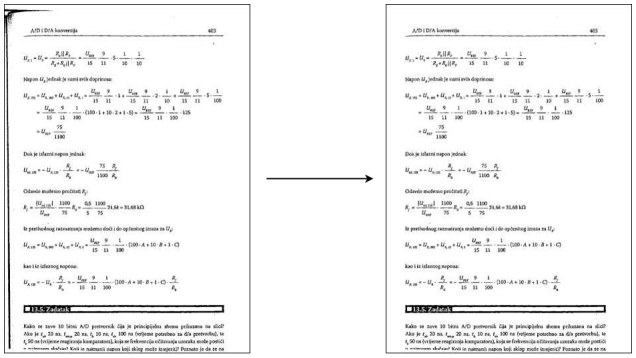
2.2. Segmentacija i predobrada

Prije nego što slika dokumenta dosegne osnovni OCR mehanizam (prepoznavanje znakova), treba proći nekoliko pripremnih koraka koji uključuju binarizaciju, segmentaciju stranice, automatsko uklanjanje nagiba, segmentaciju redaka, riječi i znakova (Doermann i Tombre, 2014).

2.2.1. Binarizacija

Cilj binarizacije je odvajanje teksta od pozadine kako bi se omogućilo izračunavanje korisnih značajki za prepoznavanje znakova. Najčešće korišteni automatski algoritam binarizacije s konstantnim pragom je Otsu algoritam. Cilj algoritma je smanjiti kombinirano širenje (varijance unutar klase) intenziteta dvije klase odvojene pragom. Algoritmi prilagodljivog praga poput Niblack algoritma i Sauvola algoritam obično su učinkovitiji u binariziranju slika dokumenata s većim šumom pomoću lokalno odabranih pragova (Doermann i Tombre, 2014).

[3] Slika je preuzeta iz (Rocchini, 2012)



Slika 2.6: Marginalno uklanjanje šuma

2.2.2. Predobrada

U predobradu spadaju koraci poput uklanjanje šuma i ispravljanje nagiba. Ovdje su navedene metode koje rade nad binarnim slikama, ali postoje metode koje rade i nad sivim slikama.

Uklanjanje šuma

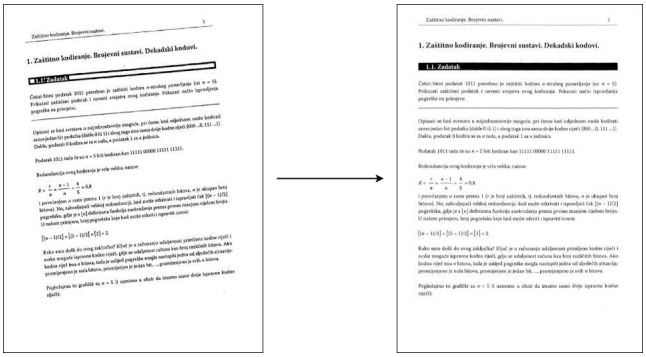
Tipičan šum kod slika (koji se viđa na slikama) je šum soli i papra. Također je poznat kao impulsni šum. Ovaj šum mogu uzrokovati oštri i nagli poremećaji u signalu slike. Predstavlja se kao bijeli i crni pikseli koji se rijetko javljaju na mjestima gdje ih se ne očekuje, tj. gdje ne bi trebali biti.

Učinkovita metoda smanjenja ove vrste šuma je medijan filter. Ako je u definiranom susjedstvu nekog piksela p više crnih nego bijelih piksela, piksel p se postavlja na crnu boju. Obrnuto, ako je više bijelih nego crnih, piksel p se postavlja na bijelu boju.

Šum spefičan kod slika skeniranih dokumenata je marginalni šum. Primjer marginalnog šuma može se vidjeti na slici 2.6. Jednostavan i učinkovit pristup uklanjanju marginalnog šuma sa slike dokumenta naveden je u (Shafait i Breuel, 2009).

Ispravljanje nagiba

Skenirani dokument na slici može biti ukošen pod proizvoljnim kutom ako se nije pazilo prilikom skeniranja. Primjer ukošenog dokumenta na slici 2.7. Uklanjanjem nagiba olakšavamo idućim koracima OCRa, konkretno, modul segmentacije stranice i modul za prepoznavanje teksta mogu pretpostaviti da tekst nije ukošen. Uklanjanje nagiba se tipično obavlja u dva koraka. U prvom se koraku procjenjuje kut nagiba cijelog dokumenta. U drugom se koraku slika zakreće za inverzni kut procjenjenog nagiba. Opisat ćemo jednu tehniku uklanjanja nagiba skeniranih slika dokumenata



Slika 2.7: Ispravljanje nagiba

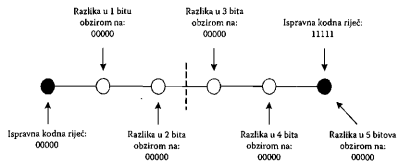
danu u (Yin, 2001) koja koristi RLSA i Houghovu transformaciju.

Prvo se slika zagladi vertikalnim RLSA postupkom. Da bi se poboljšale osnovne linije (engl. baseline) i smanjila količina podataka, što Houghovu transformaciju čini računalno skupom, izglađena slika se uzorkuje. To činimo tako da skeniramo zaglađenu sliku okomito stupac po stupac i lociramo svaki crno-bijeli prijelaz na crnom pikselu zadržavajući njegovu vrijednost kao 1 i zamjenjujući vrijednosti ostalih crnih piksela s 0. Rezultirajuća slika naglašava osnovne linije teksta. Posljednji korak je pronaći liniju koja najbolje odgovara preostalim crnim pikselima na slici (koja bi trebala biti paralelna osnovnim linijama teksta) pomoću Houghove transformacije za procjenu kuta nagiba. Ako uzmemo u obzir dva crna piksela odjednom, samo jedna linija može proći kroz dvije točke i zato samo jednom skupu parametara (p i q; u jednadžbi linije $p = xcosq + ysinq$) treba brojati učestalost pojavljivanja. Pošto nas zanima samo kut (tj. parametar q), možemo samo brojati glasove za parametar q kako bismo odredili kut nagiba teksta. Konačno, q vrijednost koja je dobila najveći broj glasova određuje kut nagiba. Detalji procjene kuta nagiba, tj. izvedbe/izračuna Houghove transformacije, mogu se naći u (Yin, 2001).

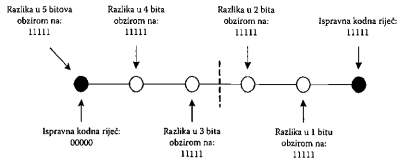
2.2.3. Segmentacija stranice

Segmentacija stranice uključuje pronalaženje i odvajanje teksta od netekstovnih područja stranice. Segmentacija stranice segmentira slike dokumenta u homogena područja kao što su tekstualna područja, grafička područja i tablice, te izdvaja tekstualna područja iz segmentiranih komponenata (Doermann i Tombre, 2014). Primjer rezultata segmentacije stranice možemo vidjeti na slici 2.8b. Izdvojena tekstualna područja dalje se proslijeđuju modulu za segmentaciju redaka teksta.

Opisat ćemo jednu metodu segmentacije stranice opisanu u (Wong et al., 1982). Algoritam radi na binarnim slikama gdje su bijeli pikseli predstavljeni s 0, a crni s 1.



Na prikazanoj slici referentna kodna riječ je 00000. Isto razmatranje možemo provesti ako kao referentnu kodnu riječ uzmemo 11111, čime dobijemo simetričnu sliku:

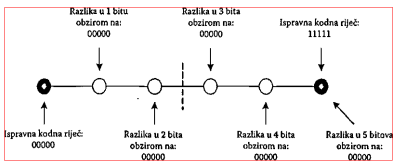


Uzevši u obzir da je vjerojatnost pojave višestruke pogreške uobičajeno manja od vjerojatnosti pojave manje pogreške, uvijek će se pretpostavljati da je došlo do minimalnog broja promijenjenih bitova. Zbog toga će se izračunati udaljenost primljene riječi do svake ispravne kodne riječi, i kao prava poslana kodna riječ bit će uzeta ona ispravna kodna riječ koja ima minimalnu udaljenost. Primjerice, ako je primljena kodna riječ 10101, računamo udaljenosti:

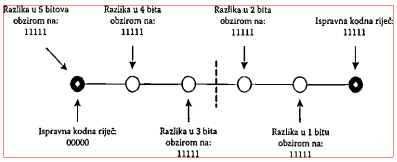
$d(10101, 00000) = 3$
$d(10101, 11111) = 2$

Dakle, pretpostavit ćemo da je poslana kodna riječ 11111, što odgovara podatku 1. Na taj način kod će pouzdano ispravljati svaku jednostruku pogrešku, i svaku dvostruku pogrešku, što smo inicijalno i utvrdili. No u slučaju trostruke pogreške – kod će zakazati. Naime, tada će udaljenost primljene riječi biti manja do one druge kodne riječi, i ispravljanje pogreške će zakazati (zbog čega smo i izrekli tvrdnju da kod pouzdano ispravlja do dvije pogreške).
Općenito vrijedi sljedeća tvrdnja: kod s minimalnom udaljenošću d između **svakih dviju** ispravnih kodnih riječi može ispraviti $\lfloor (d-1)/2 \rfloor$ pogrešaka.

2 Zbirka riješenih zadataka



Na prikazanoj slici referentna kodna riječ je 00000. Isto razmatranje možemo provesti ako kao referentnu kodnu riječ uzmemo 11111, čime dobijemo simetričnu sliku:



Uzevši u obzir da je vjerojatnost pojave višestruke pogreške uobičajeno manja od vjerojatnosti pojave manje pogreške, uvijek će se pretpostavljati da je došlo do minimalnog broja promijenjenih bitova. Zbog toga će se izračunati udaljenost primljene riječi do svake ispravne kodne riječi, i kao prava poslana kodna riječ bit će uzeta ona ispravna kodna riječ koja ima minimalnu udaljenost. Primjerice, ako je primljena kodna riječ 10101, računamo udaljenosti:

$d(10101, 00000) = 3$
$d(10101, 11111) = 2$

Dakle, pretpostavit ćemo da je poslana kodna riječ 11111, što odgovara podatku 1. Na taj način kod će pouzdano ispravljati svaku jednostruku pogrešku, i svaku dvostruku pogrešku, što smo inicijalno i utvrdili. No u slučaju trostruke pogreške – kod će zakazati. Naime, tada će udaljenost primljene riječi biti manja do one druge kodne riječi, i ispravljanje pogreške će zakazati (zbog čega smo i izrekli tvrdnju da kod pouzdano ispravlja do dvije pogreške).
Općenito vrijedi sljedeća tvrdnja: kod s minimalnom udaljenošću d između **svakih dviju** ispravnih kodnih riječi može ispraviti $\lfloor (d-1)/2 \rfloor$ pogrešaka.

(a) **(b)**

Slika 2.8: Segmentacija stranice

RLSA zaglađivanje primjenjuje se horizontalno i vertikalno. Slika 2.9 prikazuje primjer. Izvorna slika prikazana na slici2.8a pretvara se u horizontalno zaglađene (a) i vertikalno zaglađene (b) slike. RLSA uzima logično I ovih horizontalno i vertikalno zaglađenih slika kako bi generirao sliku 2.9c. Dodatno horizontalno zaglađivanje pomoću RLSA daje konačni rezultat segmentacije prikazan na 2.9d.

RLSA, u odnosu na većinu drugih algoritama za segmentaciju stranice, segmentira stranicu na retke teksta umjesto regije teksta.

Algoritam segmentacije stranice pomoću RLSA[4]

- RSLA postupak prvo provedemo horizontalno, a zatim vertikalno da bi se dobile dvije različite slike
- Dvije slike kombiniraju se uporabom logičkog operatora I
- Izglađena završna bitmapa dobiva se ponovnim zaglađivanjem kombinirane slike u horizontalnom smjeru
- Povezane komponente u završnoj bitmapi odgovaraju segmentima na orginalnoj slici

Nakon segmentacije stranice, slijedi klasifikacija izdvojenih područja. U svrhu prepoznavanja teksta dovoljno je samo moći razlikovati tekstualne komponente od netekstualnih, kako bi tekstualna područja mogli dalje proslijediti idućim modulima.

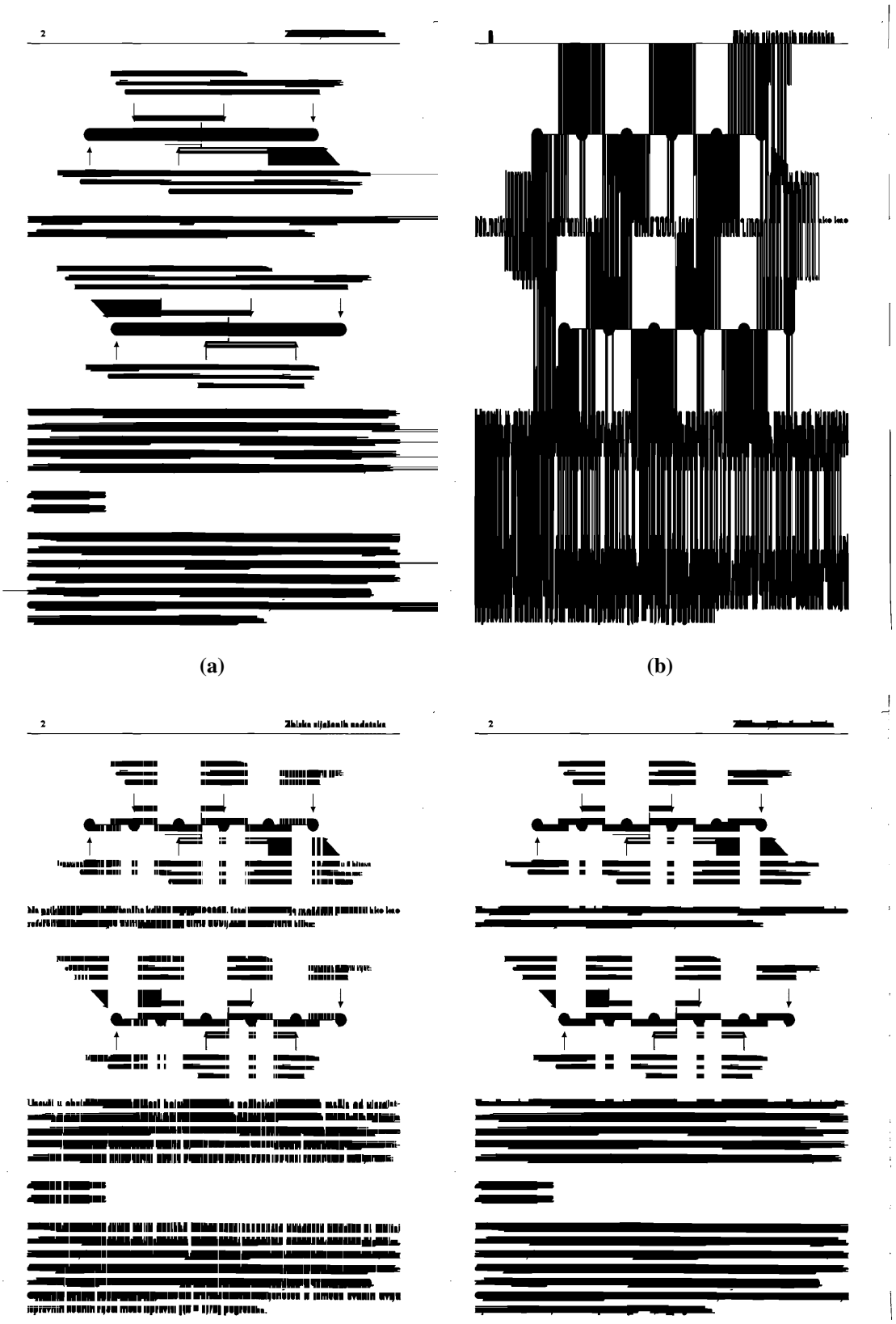
2.2.4. Segmentacija redaka, riječi i znakova

Segmentacija redaka teksta problem je podjele slike tekstualne regije na slike redaka teksta. Segmentacija riječi problem je podjele slike retka teksta na slike riječi. Segmentacija znakova problem je podjele slike riječi ili slike retka teksta u slike znakova (Doermann i Tombre, 2014).

Segmentacija redaka

Korištenje horizontalne projekcije vjerojatno je najlakši način za segmentiranje redaka. Ova je metoda idealna kada su retci dobro odvojeni i ne preklapaju se ili ne dodiruju jedan drugog. U tom slučaju, projekcija će sadržavati vrijednosti različite od nule za linije skeniranja koje prelaze redak teksta. Razmaci između redaka imat će vrijednosti nula, čime se učinkovito označavaju početak i završetak granica redaka teksta (Doermann i Tombre, 2014).

[4] Algoritam je preuzet iz (Shafait, 2011).



(a) **(b)**

(c) **(d)**

Slika 2.9: Segmentacija stranice pomoću RLSA

Male vrijednosti mogu biti prisutne u prazninama ako je prisutan neki šum. U tom se slučaju može primijeniti granična vrijednost za uklanjanje lažno pozitivnih redaka. Horizontalna projekcija najprikladnija je za tiskani tekst. Tiskani tekst dosljedniji je kad je riječ o održavanju praznina između redaka i održavanju redaka horizontalno ravnim, u odnosu na rukom pisani tekst. (Doermann i Tombre, 2014).

Segmentacija riječi

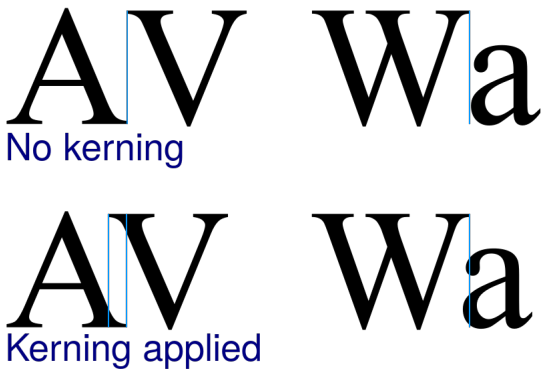
Segmentacija riječi je, kao i segmentacija redaka, jednostavna za tiskani tekst. Vertikalnom projekcijom izračunamo udaljenosti između susjednih znakova. Procjenimo graničnu vrijednost koja dijeli izračunate udaljenosti na udaljenosti između susjednih znakova iste/jedne riječi i na udaljenosti zadnjeg znakova jedne riječi (ili znaka kraja rečenice, npr. ".") i prvog znakova iduće riječi. Procjenjenom graničnom vrijednošću grupiramo znakova u riječi.

Segmentacija znakova

Slike znakova mogu se prepoznati pomoću praznog prostora (vertikalna projekcija) (Breuel, 2010) i analize povezanih komponenata (Parker, 2010) kada su dobro odvojeni. Problemi kod pristupa vertikalne projekcije nastaju sa znakovima u kurzivu i sa sljubljenim znakovima (kad jedna vertikalna linija prolazi kroz 2 znaka) kao što možemo vidjeti na slici 2.10. Kod pristupa povezanih komponenata znakovi poput "i" ili "j" te dijakritički znakovi (č, ć, đ, š i ž) razdvojeni su u dvije komponente. Povezane komponente jednog znakova se mogu grupirati prije ili poslije (Radošević, 1996) prepoznavanja pojedinog znaka. Oba pristupa ne uspijevaju segmentirati znakove koji se dodiruju i slomljene/razlomljene znakove (znakovi koji su jedna komponenta, ali su zbog npr. binarizacije razlomljeni u dvije ili više komponenti).

Općenito je segmentacija znakova vrlo izazovan problem i zahtijeva povratne informacije od modula za prepoznavanje kako bi se poboljšala točnost. Tipično će se segmentirati riječ u duljine koje su puno kraće od prosječne duljine znaka i koristiti pravila rekombinacije za spajanje n susjednih atomskih segmenata u hipoteze znakova (Doermann i Tombre, 2014).

Ako se radi o fontu kod kojeg su sva znakova jednake širine (engl. monospaced font, fixed-pitch), heuristike poput prosječne širine znakova mogu se koristiti kao povratna informacija za rekurzivno odvajanje slika znakova (Casey i Lecolinet, 1996). U slučaju kada znakovi nisu jednake širine, heuristike poput maksimalne širine znaka mogu se koristiti kod odabira broja susjednih atomskih segmenata koji se spajaju u



Slika 2.10: Sljubljivanje znakova

hipotezu znaka (nema smisla na prepoznavanje slati spoj susjednih segmenata koji premašuje maksimalnu širinu znaka) (Breuel, 2001).

2.2.5. Segmentacija znakova

Kao što je bilo više puta rečeno, segmentacija znakova može se poboljšati kombinirajući je s modulom za prepoznavanje. Ovdje ćemo detaljnije objasniti kako to funkcionira.

Značajke za pronalaženje segmentacijskih točaka

Kada postoje znakovi koji se dodiruju, značajke poput lokalnih minimuma u profilu vertikalne projekcije mogu se koristiti za pronalaženje optimalnih mjesta granica znakova (Doermann i Tombre, 2014; Parker, 2010).

Srodna tehnika izračunava cijenu prekida (engl. break cost). Vrijednost cijene prekida definira se kao broj crnih piksela u stupcu koji također imaju crnog susjeda u istom retku prethodog stupca, i izračunava se za svaki stupac. Stupci u kojima je cijena prekida mala kandidati su za mjesta na kojima se dijeli povezana komponenta (Parker, 2010).

Segmentacija zakrivljenim rezom prije poteza (engl. curved pre-stroke cut, CPSC) dopušta zakrivljene rezove koji segmentiraju znakove riječi (slika 2.11). CPSC algoritam procjenjuje veliki skup zakrivljenih rezova kroz sliku riječi pomoću dinamičkog programiranja i odabire mali "optimalni" podskup rezova za segmentaciju. Staza reza određuje njegov trošak. Trošak je obrnuto proporcionalan vertikalnosti staze i obrnuto proporcionalan broju crnih piksela (piksela znaka) kroz koje rez prolazi. Rez je

Slika 2.11: CPSC segmentacija tiskano pisanog teksta

bolji što mu je trošak manji. Algoritam dinamičkog programiranja koji se koristi za pronalaženje optimalnih rezova može se naći u radu (Breuel, 2001).

Segmentacija temeljena na prepoznavanju

Sekvencijalna segmentacija slika znakova bez povratne informacije od prepoznavanja ranjiva je na znakove koji se dodiruju i razlomljene znakove te značajno utječe na pogreške kod prepoznavanja znakova. Segmentacija temeljena na prepoznavanju stvara višestruke hipoteze segmentacije slike riječi i odabire optimalnu hipotezu pomoću rezultata podudaranja (povratne informacije od modula za prepoznavanje). Razvijene su metode koje pružaju prekomjernu segmentaciju slika riječi (Doermann i Tombre, 2014). Osnovni koraci su:

1. Pronađite dovoljno točaka segmentacije da bi se uključile sve ispravne točke segmentacije. Može se uzeti unija točaka segmentacije više metoda (npr. razmak, povezane komponente i analiza profila projekcije).

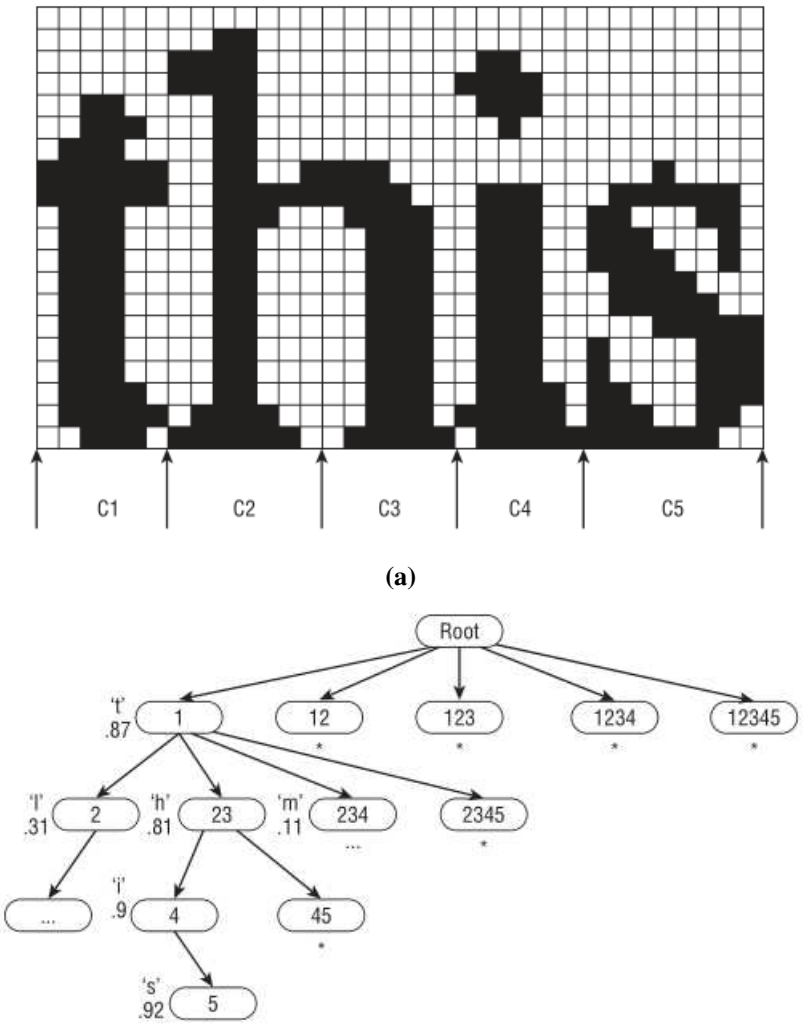
2. Odaberite optimalan podskup točaka prekomjerne segmentacije (optimalnu segmentaciju) koji minimizira pogrešku podudaranja dobivenog niza znakova pomoću dinamičkog programiranja.

Optimalna segmentacija segmentacije temeljene na prepoznavanju

Sljedeći primjer pristupa pronalaska optimalne segmentacije je preuzet iz (Parker, 2010).

Pronađemo lokacije rezova (točke segmentacije) slike riječi jednom od prethodno spomenutih metoda. Tada bi se pikseli između svaka dva položaja reza smatrali slikom znaka i pokušalo bi se prepoznavanje. Skup ispravnih segmentacija bit će odabran skup izrezanih položaja za koje su prepoznati svi znakovi (i koji imaju najveću vjerojatnost da su točni). Razmotrite sliku riječi na slici 2.12[5]. Položaji reza određeni troškovima prekida prikazani su na slici kao strelice koje pokazuju na položaj na kojem treba rezati; ima ih šest, od kojih bi bilo koja (ili bilo koja uzastopna kombinacija) mogla

[5]Slike su preuzete iz (Parker, 2010).



Slika 2.12: Točke segmentacije riječi i stablo odluke

biti stvarni znak. Sljedeći korak u procesu segmentacije je stvaranje stabla odluke koji sadrži moguća grupiranja regija na slici. Svaka susjedna regija odgovara čvoru u stablu, kao što je prikazano na slici 2.12b. Uz svaki čvor također je povezan znak (što je klasifikacija regije) i mjera vjerojatnosti. Najvjerojatnija staza kroz stablo bit će odabrana kao ispravna segmentacija i klasifikacije će stoga biti odmah dostupne.

Kada čvor (skup susjednih regija) može biti prihvaćen (klasificiran sa značajnom vjerojatnošću), klasifikacija i vjerojatnost pohranjuju se u relevantni čvor. Tek tada se podređeni čvorovi obrađuju; djeca su moguće kombinacije preostalih čvorova (regija). Čvorovi koji daju vrlo lošu klasifikaciju ne vrednuju se dalje, barem u početku. Ako se dogodi da niti jedan put kroz stablo ne rezultira dobrom klasifikacijom, mogu se procijeniti manje vjerojatne grane. Stablo se može konstruirati kako je opisano ili se

može izraditi rekurzivna dekompozicija. Skup svih mogućih rekurzivnih poziva čini stablo, a jedino će se zadržati najbolja segmentacija, a ostale će se odbaciti čim se pronađe bolja. Ovaj pristup ovisi o preciznosti metode prepoznavanja znakova koja se primjenjuje na rezana područja.

U (Parker, 2010) piše da će biti odabran skup izrezanih položaja za koje su prepoznati svi znakovi i koji imaju najveću vjerojatnost da su točni, ali nije rečeno kako se određuje vjerojatnost svih znakova odjednom. Čini se logičnim uzeti umnožak pojedinačnih vjerojatnosti. Što ako jedan podskup ima više prepoznatih znakova od drugog? Je li i onda uredu uspoređivati umnoške vjerojatnosti? U radu (Burges et al., 1992) piše da bi se u tom slučaju trebalo "prikladno normalizirati", ali nije rečeno i kako. Pretpostavljamo da je bez pristupa drugih informacija (npr. vjerojatnosti pojavljivanja riječi određene duljine) u redu samo pomnožiti vjerojatnosti.

2.2.6. Normalizacija

Većina klasifikatora koji se koriste u modulu za prepoznavanje na ulazu očekuju fiksni dimenzijski ulaz. Razlike u veličini znakova mogu povećati varijancu oblika karaktera unutar klase. Stoga je normalizacija slike znakova važan korak u većini algoritama za odabir značajki. Najjednostavnija normalizacija slike znaka je normalizacija graničnim okvirom (Breuel, 2010).

Pristupi klasificiranja temeljeni na učenju mogu učinkovito kombinirati značajke izvučene iz sirove slike kao i normaliziranu sliku kako bi poboljšali performanse u odnosu na upotrebu bilo kojeg skupa značajki (Doermann i Tombre, 2014).

Normalizacija graničnim okvirom

Normalizacijom graničnim okvirom želimo sliku znaka dobivenu segmentacijom riječi transformirati u novu sliku unaprijed određenih dimenzija u kojoj znak dodiruje sva četiri stranice slike.

2.3. Izolirano prepoznavanje znakova

U ovom odjeljaku opisujemo načine prepoznavanja pojedinog znaka te metode prikupljanja podataka za treniranje.

2.3.1. Izdvajanje značajki

Rano je prepoznata potreba za diskriminacijskim značajkama koje mogu pomoći mehanizmu za prepoznavanje da učinkovito razlikuje slike znakova. Stoga je, osim dizajna i treninga klasifikatora, dizajn i primjena tehnika ekstrakcije značajki bio važan fokus istraživanja u izoliranom prepoznavanju znakova (Doermann i Tombre, 2014).

Unatoč mnogim razlikama između postojećih jezika u svijetu, znakovi se u većini jezika prikazuju kao linijski crteži. To je omogućilo brži napredak u razvoju pristupa prepoznavanja nego što bi to inače bilo moguće. Značajke histograma su najčešće korištena obitelj značajki za prepoznavanje znakova. Značajke histograma na razini piksela slike znaka dobivaju se rastavljanjem signala slike oko piksela u nekoliko orijentacija pomoću usmjerenih transformacija (operator gradijenta, konkavnost itd.). Značajke cijele slike obično se dobivaju uzorkovanjem slike znaka i horizontalno i vertikalno u određenim intervalima i računanjem značajki histograma na tim odabranim mjestima. Značajke histograma učinkovit su alat za opisivanje strukture linija znakova (Doermann i Tombre, 2014).

GSC značajke

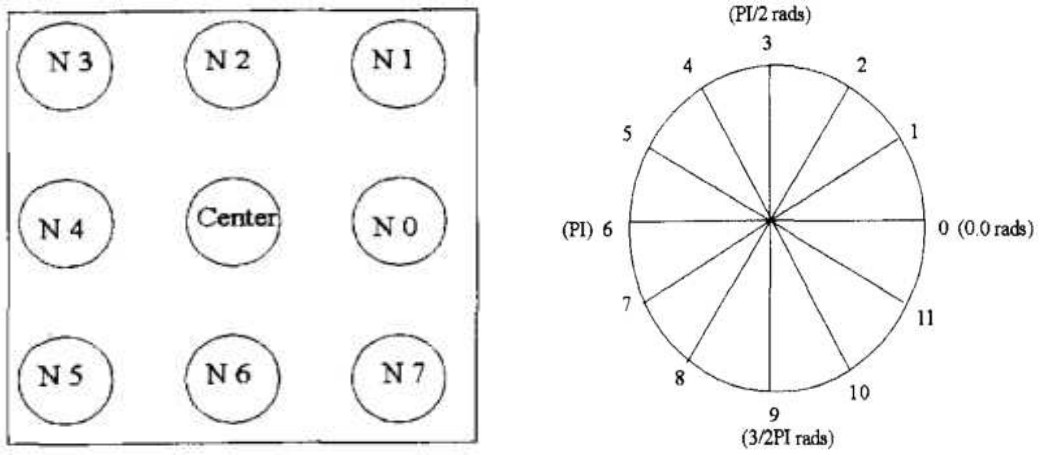
Opis GSC značajki je izvučen iz (Doermann i Tombre, 2014; Favata i Srikantan, 1996). Značajke gradijenta, strukture i konkavnosti (GSC) su primjer značajki histograma za binarne slike znakova. Slika znaka podijeljena je na četiri dijela koje ćemo dalje zvati podslike znaka. Iz svake podslike znaka izračunavaju se tri vrste značajki (gradijentna, strukturna i konkavna). Značajke gradijenta izračunavaju se primjenom Sobelovih operatora gradijenta prikazanih u jednadžbi 2.1:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.1}$$

na cijelu sliku znaka. Gradijent svakog piksela kvantiziran je u 12 orijentacija: $30 \times i^\circ$ $(i = 0, 1, \ldots, 11)$. Ukupno 12 značajki definirano je kao

$$f_{gi} = \frac{\text{\#crnih piksela u podslici znaka s gradijentnom orijentacijom } i}{\text{\#piksela u podslici znaka}} \tag{2.2}$$

Brojnik u jednadžbi 2.2 ne uključuje piksele gradijenata magnitude nula jer su njihove orijentacije nedefinirane.



(a) Definicija 8 povezivosti **(b)** 12 jednakih područja uzorkovanja

Slika 2.13

Strukturne značajke zahvaćaju određene obrasce ugrađene u gradijentnoj matrici. Ti su uzorci "mini potezi" slike. Skup od 12 pravila primjenjuje se na svaki piksel. Ova pravila djeluju na osam najbližih susjeda piksela. Svako pravilo ispituje određeni obrazac susjednih piksela za dopuštene raspone gradijenta. Na primjer, pravilo 1 navodi da ako susjed (N 0) piksela (vidi sliku 2.13[6] za definicije povezanosti i gradijenta) ima raspon gradijenta 2,3,4, a susjed (N 4) ima raspon gradijenta 2,3,4, tada je pravilo zadovoljeno. Trenutni skup pravila, naveden u tablici 2.1[7], uključuje četiri vrste uglova i osam vrsta linija. Za svaku podsliku znaka definirano je ukupno 12 strukturnih značajki. Svaka strukturna značajka definirana je kao

$$f_{si} = \frac{\text{\#crnih piksela strukturne klase } i \text{ u podslici znaka}}{\text{\#piksela u podslici znaka}} \tag{2.3}$$

Značajke konkavnosti razlažu pozadinski signal slike (bijeli pikseli) u različite orijentacije. Ove orijentacije opisuju konkavnost poteza znaka. Koraci za izračunavanje značajki konkavnosti su sljedeći. Prvo skenirajte sliku znakova red po red slijeva nadesno i zdesna nalijevo i stupac po stupac odozgo prema dolje i odozdo prema gore. Tijekom skeniranja, pratite sve skenirane bijele piksele. Kad naiđete na prvi crni piksel, stanite, i krenite u novi red/stupac.

U sljedećem koraku, značajke histograma uzimaju se na isti način kao i kod gradijentnih i strukturnih značajki pomoću sljedećeg izračuna:

[6]Slika preuzeta iz (Favata i Srikantan, 1996)

[7]Tablica definicija strukturnih značajki je preuzeta iz (Favata i Srikantan, 1996).

Pravilo	Opis	Susjed 1 (Raspon)	Susjed 2 (Raspon)
1	Horizontalna linija, tip 1	N 0 (2,3,4)	N 4 (2,3,4)
2	Horizontalna linija, tip 2	N 0 (8,9,10)	N 4 (8,9,10)
3	Vertikalna linija, tip 1	N 2 (5,6,7)	N 6 (5,6,7)
4	Vertikalna linija, tip 2	N 2 (1,0,11)	N 6 (1,0,11)
5	Dijagonalno dizanje, tip 1	N 5 (4,5,6)	N 1 (4,5,6)
6	Dijagonalno dizanje, tip 2	N 5 (10,11,0)	N 1 (10,11,0)
7	Dijagonalno spuštanje, tip 1	N 3 (1,2,3)	N 7 (1,2,3)
8	Dijagonalno spuštanje, tip 2	N 3 (7,8,9)	N 7 (7,8,9)
9	Kut 1	N 2 (5,6,7)	N 0 (8,9,10)
10	Kut 2	N 6 (5,6,7)	N 0 (2,3,4)
11	Kut 3	N 4 (8,9,10)	N 2 (1,0,11)
12	Kut 4	N 6 (1,0,11)	N 4 (2,3,4)

Tablica 2.1: Skup pravila strukturnih značajki

$$f_{ci} = \frac{\text{\#broj bijelih piksela u podslici znaka do kojih se može doći samo prilikom skeniranja u orijentaciji } i}{\text{\#piksela u podslici znaka}} \tag{2.4}$$

Slično tome, značajka rupe definirana je kao

$$f_{hole} = \frac{\text{\#broj bijelih piksela u podslici znaka do kojih se ne može doći iz bilo koje orijentacije}}{\text{\#piksela u podslici znaka}} \tag{2.5}$$

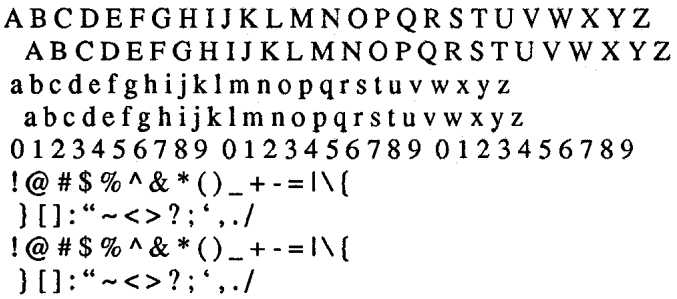
2.3.2. Prepoznavanje znakova

Bilo koji klasifikator/klasifikacijski model strojnog učenja može se koristiti za prepoznavanje znakova (kNN, slučajna šuma, SVM, neuronska mreža, itd.). U ulaz modela možemo dovoditi piksele slike znaka kao značajke, ili možemo izračunati/izvući značajke iz slike (objasnjeno u pododjelju 2.3.1) i njih dovesti na ulaz našeg modela.

2.3.3. Modeliranje retka teksta

Na slici 2.14 vidimo da se dio znakova "S", "h" i "i" proteže iznad glavnog dijela, tj. iznad srednje linije (engl. ascender), dio znakova "p" se proteže ispod osnovne linije (engl. descender) dok su znakovi "n" i "x" između osnovne i srednje linije. Tim bi informacijama klasifikator znakova mogao razlikovati mala od velikih slova koja

Slika 2.14: Modeliranje retka teksta



Slika 2.15: Slika za treniranje

jednako izgledaju (npr. "s" i "S") ili recimo "9" od "g" (Breuel, 2010). U poglavlju 3 ćemo objasniti kako pravilno kodirati te informacije u značajke.

2.3.4. Podaci za treniranje

Osim u posebnim okolnostima (poput stranica koje sadrže matematiku ili više jezika), ne bi trebalo biti više od 128 različitih znakova koje treba identificirati. Budući da je računalo prilično brz stroj, moguće je provjeriti dolaznu/ulaznu sliku znaka sa svim mogućim znakovima i klasificirati ju kao najbolje podudaranje. Da bi to bilo moguće, program mora biti osposobljen za prepoznavanje svih mogućih znakova. Jednostavan način kako to učiniti je pružiti programu predložak za svaki znak, dok složeniji pristup uključuje obradu/anotaciju slika skeniranih dokumenata kojima raspolažemo (Parker, 2010).

Pružanje predložaka za svaki znak

Za ovaj pristup potrebna nam je standardna tekstualna slika koja sadrži sve znakove koji se prepoznaju (koje želimo prepoznavati) u poznatom redoslijedu. Na taj se način program može sam osposobiti; poznat je prvi znak na tekstualnoj slici, a program "čita"/izvlači prvi znak sa slike i koristi se kao primjer svoje klase. Slika 2.15[8] prikazuje primjer slike za treniranje koju sustav može koristiti (Parker, 2010).

[8] Slika je uzeta sa web stranice www.wiley.com/go/jrparker.

3. Prototipna programska implementacija sustava

Za implementaciju našeg prototipa odlučili smo se koristiti programski jezik Python. OCR sustav ovisi o sljedećim paketima/modulima/razredima/funkcijama:

- paket numpy - koristi se za reprezentaciju slike (2D binarno numpy polje), za osnovnu manipulacija nizom/poljem
- modul scipy.ndimage - koristi se za obradu slika
- funkcija numba.njit - koristi se za kompilaciju često pozivajućih funkcija koje se ne mogu napisati pomoću numpy funkcija
- paket scikit-learn - koristi se za klasifikaciju znakova
- modul skimage.filters.thresholding[1] - koristi se za binarizaciju slike
- funkcija skimage.filters.transform.resize - koristi se za normalizaciju slike
- funkcija matplotlib.pyplot.imread - koristi se za čitanje slike iz datoteke u numpy polje

OCR sustav je podijeljen u četiri glavna modula: `image_processing`, `preprocessing`, `layout_analysis`, `text_line_recognition`.

U modulu `image_processing` nalaze se implementacije algoritama za obradu slika danih u odjeljku 2.1. Ostatak glavnih modula implementira blokove dijagrama OCR sustava danog u poglavlju 2.

Modul `preprocessing` implementira blokove *binarizacija* i *predobrada*. Pruža funkcionalnost binarizacije slike, ispravljanje nagiba, te uklanjanja šuma.

Modul `layout_analysis` implementira blokove *segmentacija stranice* i *segmentacija retka teksta*. Pruža implementacije lociranja i izvlačenja redaka teksta binarne slike skeniranog dokumenta.

[1] skimage je sinonim za scikit-image

Modul `text_line_recognition` implementira blokove *segmentacija riječi*, *segmentacija znakova* i *prepoznavanje znakova*. Pruža implementacije prepoznavanja teksta binarne slike retka teksta.

Moduli `label_text_lines` i `label_characters` su grafička korisnička sučelja za označavanje redaka teksta na stranici i označavanje znakova u retku teksta te služe za izgradnju skupa podataka. Za izradu GUIa koristio se Pythonov paket Tkinter.

3.1. Glavni moduli sustava

Ovaj odjeljak dokumentira glavne module sustava. U tekstu se spominje termin "lokacija" (engl. location) nekog objekta (npr. retka teksta ili znaka) unutar slike i odnosi se na pravokutni prozor čije su stranice paralelne sa stranicama slike.

3.1.1. Modul `image_processing`

Označavanje povezanih komponenata je implementirano za 4-povezana piksela. Implementirana je segmentacija projekcijom za horizontalni i vertikalni smjer. Houghova transformacija nije implementirana. Sve funkcije modula `image_processing` rade s binarnom slikom i pretpostavljaju da 1 (`true`) predstavlja crni piksel.

3.1.2. Modul `preprocessing`

Funkcija `binarize` na ulazu prima sivu sliku ili sliku u boji (rgb), a na izlazu vraća binarnu sliku gdje 1 (`true`) predstavlja crni piksel. Po defaultu koristi Otsu algoritam za binarizaciju slike, ali to se može promjeniti pozivom funkcije `binarize` s dodatnim argumentom `threshold` pomoću kojeg se izračunava prag binarizacije slike. Parametar `threshold` mora biti funkcija koja prima sivu sliku, a vraća jednu vrijednost za prag ili matricu pragova za svaki piksel slike.

Funkcija `correct_skew` na ulazu prima binarnu sliku dokumenta, a na izlazu vraća sliku dokumenta s ispravljenim nagibom. Implementirana je tehnika uklanjanja nagiba objašnjena u pododjeljku 2.2.2.

3.1.3. Modul `layout_analysis`

Definirano je sučelje `TextLineSegmenter` za segmentaciju redaka teksta binarne slike skeniranog dokumenta.

```
class TextLineSegmenter:
    """ Segments text lines from page image """

    def locate_text_lines(self, page):
        """ page → text line locations """
        raise NotImplementedError

    def segment_text_lines(self, page):
        """ page → text lines """
        line_locations = self.locate_text_lines(page)
        return [page[loc] for loc in line_locations]
```

Sučelje `TextLineSegmenter` definira apstraktnu metodu `locate_text_lines` koja prima binarnu sliku dokumenta, a vraća lokacije redaka teksta na dobivenoj slici.

U nastavku slijede implementacije sučelja `TextLineSegmenter`.

HorizontalProjection

Klasa `HorizontalProjection` pretpostavlja da se na slici dokumenta nalaze samo redovi teksta, ne i druge, ne tekstualne, komponente (npr. slika).

RunLengthSmearing

Klasa `RunLengthSmearing` implementira *Algoritam zaglađivanja dužine trčanja (RLSA)* dan u pododjeljku 2.2.3.

3.1.4. Modul `text_line_recognition`

Definirana su sučelja `TextLineRecognizer` za prepoznavanje teksta binarne slike retka teksta, `CharacterClassifier` za klasificiju znaka i `TextLineCharFeatureExtractor` za izvlačenje značajki znaka retka teksta.

```
class TextLineRecognizer:
    """ Recognizes text from text line image """

    def recognize_text_line(self, text_line):
        """ text_line → str """
        raise NotImplementedError

class CharacterClassifier:
    """ Classifies character from input features """

    def classify(self, x, return_prob=False):
        """ x → char[, probability] """
        raise NotImplementedError

class TextLineCharFeatureExtractor:
    """ Extracts text line character features """

    def set_text_line(self, text_line):
        raise NotImplementedError

    def extract_features(self, char, char_location):
        raise NotImplementedError
```

Sučelje TextLineCharFeatureExtractor

Sučelje `TextLineCharFeatureExtractor` omogućava modeliranje retka teksta (objašnjeno u pododjeljku 2.3.3). Metodu `set_text_line` potrebno je pozvati sa slikom retka teksta iz čijih znakova želimo izvući značajke. Značajke znaka dobivaju se pozivanjem metode `extract_features` sa slikom znaka i položajem znaka unutar slike retka teksta. Položaj znaka je potreban kako bi se mogle izračunati značajke vezane uz položaj znaka u odnosu na redak teksta (npr. nalazi li se dio znaka ispod osnovne linije). Slika znaka je potrebna jer neke metode segmentiranja znakova (npr. CPSC, pododjeljka 2.2.5) ne segmentiraju znak vertikalnim linijama inače bi samo lokacija znaka bila dovoljna za izračunavanje značajki (tada bi mogli izvući sliku znaka iz slike retka teksta dobivenog pozivom metode `set_text_line`).

Razred CharFeatureExtractor

Razred `CharFeatureExtractor` je implementacija sučelja `TextLineCharFeatureExtractor` koja izračunava značajke samo na temelju slike znaka. Funkciju koja izračunava značajke iz slike znaka prima preko kontruktora.

Razred ExtractEncodeResize

Razred `ExtractEncodeResize` je implementacija sučelja `TextLineCharFeatureExtractor` koja uz značajke slike znaka kodira položaj y i visinu znaka. Položaj y i visina se kodiraju tako da se podijele s visinom malih slova (engl. x-height)[2]. Visina malih slova se izračuna kao razlika položaja y osnovne i srednje linije. Položaj y osnovne linije je procijenjen kao medijan donjih granica graničnih kutija znakova retka teksta, a položaj y srednje linije je procijenjen kao medijan gornjih granica graničnih kutija znakova retka teksta. Implementacija pretpostavlja da je u tekstu više od pola znakova "malo" (engl. lower case). Značajke samog znaka i značajke znaka u retku tekstu se konkateniraju. Funkciju koja izračunava značajke iz slike znaka prima preko kontruktora.

Razred SklearnClassifier

Razred `SklearnClassifier` je implementacija sučelja `CharacterClassifier` koja preko kontruktora prima klasifikator paketa `scikit-learn` i koristi ga za klasifikaciju znaka.

SegmentationWithCharRecognition

Apstraktni razred `SegmentationWithCharRecognition` implementira sučelje `TextLineRecognizer`, tj. implementa metodu `recognize_text_line` segmentacijom retka teksta na znakove, i klasifikacijom svakog znaka. Ne daje implementaciju metode `locate_characters` kojom se segmentiraju znakovi retka teksta. Klasifikaciju znakova izvrši izvlačenjem značajki svakog znaka i slanjem tih značajki klasifikatoru znaka. Klasifikator znakova (implementacija sučelja `CharacterClassifier`) i izvlačivača značajki (implementacija sučelja `TextLineCharFeatureExtractor`) prima preko kontruktora.

[2]Ovo dijeljenje je bitno jer za 2 retka različitih veličina fontova položaj y i visina za isti znak će biti različiti. Dijeljenjem s x-height dobivamo relativni položaj y i visinu.

VProjectionSegmentationWithCharRecognition

Razred `VProjectionSegmentationWithCharRecognition` nasljeđuje razred `SegmentationWithCharRecognition` i segmentira/locira znakove vertikalnom projekcijom.

ConcompSegmentationWithCharRecognition

Razred `ConcompSegmentationWithCharRecognition` nasljeđuje razred `SegmentationWithCharRecognition` i pronalazi povezane komponente u slici retka teksta. Povezane komponente koje su jedna iznad druge (poput komponeti slova "i") grupiraju se u jednu zajedničku komponentu. Izvučene povezane komponente tretira kao znakove.

CharRecognitionBasedSegmentation

Apstraktni razred `CharRecognitionBasedSegmentation` implementira sučelje `TextLineRecognizer`. Ovaj razred se koristi tehnikom segmentacije temeljene na prepoznavanju (objašnjeno u pododjeljku 2.2.5). Implementira stablo odluke rekurzivnom dekompozicijom. Dinamičko programiranje je postignuto tehnikom memoizacije. Lokacije riječi retka teksta su pronađene i za svaku riječ poziva se apstraktna metoda `find_segmentation_cuts` koja vraća listu točaka segmentacije. Za svaku riječ i njegovu pripadnu listu točaka segmentacije stablom odluke pretražuje se najbolji podskup točaka segmentacije. Apstrahirana je i segmentacija znaka pomoću dvije točke segmentacije metodom `segment_char`. Ovime smo omogućili/postigli da graf segmentacije radi neovisno o reprezentaciji "točake" segmentacije, tj. znak može biti segmentiran s dva vertikalna pravca, ali i s dva zakrivljena reza, ovisno o konkretnoj implementaciji.

U odnosu na razred `SegmentationWithCharRecognition` koji segmentira znakove retka teksta neovisno o njihovom prepoznavanju, razred `CharRecognitionBasedSegmentation` traži optimalnu segmentaciju znakova u riječi, koja maksimizira prepoznavanje znakova. Oba razreda segmentiraju redak teksta na riječi (postupak objašnjen u pododjeljku 2.2.4).

CPSCCharRecognitionBasedSegmentation

Razred `CPSCCharRecognitionBasedSegmentation` nasljeđuje razred `CharRecognitionBasedSegmentation` i implementira segmentaciju zakriv-

ljenim rezom prije poteza opisanu u pododjeljku 2.2.5.

BreakCostCharRecognitionBasedSegmentation

Razred `BreakCostCharRecognitionBasedSegmentation` nasljeđuje razred `CharRecognitionBasedSegmentation` i pronalazi točke segmentacije pomoću cijene prekida (opisana u pododjeljku 2.2.5).

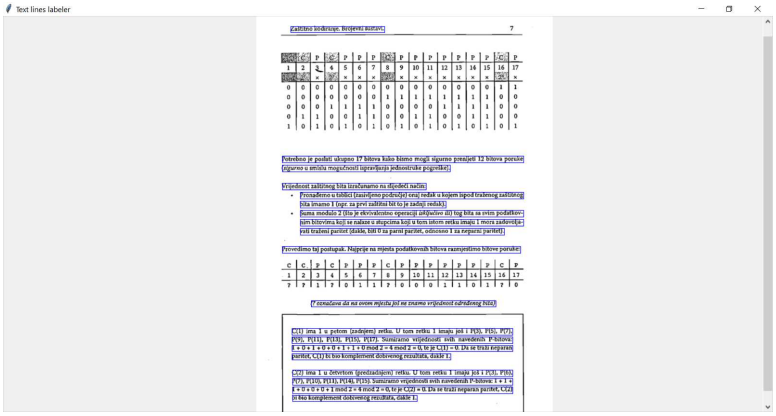
3.1.5. Primjer OCRa

Dajemo primjer jednostavnog, ali apstraktnog, OCRa izgrađenog pomoću danih sučelja koji na ulazu prima/očekuje sliku odskenirane stranice teksta/dokumenta, a na izlazu daje prepoznati tekst.

```
class SimpleScannedPageOCR:
    """ page → text """

    def __init__(self, line_segmenter, line_recognizer):
        self.segment_lines = line_segmenter.segment_text_lines
        self.recognize_line = line_recognizer.recognize_text_line

    def recognize_text(self, page):
        page = binarize(page)
        page = correct_skew(page)

        recognized_text = ""
        for line in self.segment_lines(page):
            line_text = self.recognize_line(line)
            recognize_text += line_text + '\n'

        return recognized_text
```

3.2. Alat za označavanje

Moduli `label_text_lines` i `label_characters` su grafička korisnička sučelja za označavanje redaka teksta na stranici i označavanje znakova u redku teksta.



Slika 3.1: Označeni redovi teksta

Slike skeniranih stranica knjige stavimo u jedan direktorij, npr. direktorij `book`. Pretpostavlja se da su stranice unaprijed binarizirane i da im je nagib ispravljen. Program `label_text_lines` prilikom pokretanja očekuje putanju do slike stranice kao argument naredbenog retka. Program će unaprijed za nas/automatski prilikom pokretanja potražiti retke teksta pomoću razreda `RunLengthSmearing` i označiti ih pravokutnicima (slika 3.1).

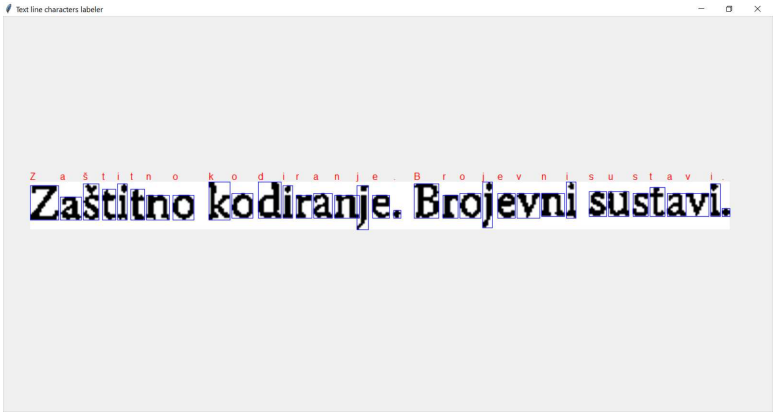
Program `label_text_lines` trenutno nudi označavanje pravokutnikom[3]. Ako pravokutnikom prođemo preko jednog postojećeg pravokutnika, izbrisat ćemo ga. Ako prođemo preko više od jednog, spojit će označene pravokutnike. Ako označimo dio slike na joj se ne nalazi niti jedan pravokutnik, program će izvršiti segmentaciju redaka horizontalnom projekcijom na taj dio slike i označiti pronađene segmente/retke minimalnim ograničavajućim pravokutnikom. Dodatno se nudi mogućnost ispravljanja svakog pojedinačnog pravokutnika pomicanjem stranice ili vrhova.

Nakon što smo završili s označavanjem redaka zatvaranjem programa spremaju se ograničavajuće kutije (engl. bounding box) označenih redaka u tekstualnu datoteku istim imenom kao i slika stranice s kojom smo pokrenuli program i stvara se direktorij s istim imenom u koji se stvaraju slike redaka teksta[4]. Ako ponovno pokrenemo program s istom slikom, program će "vidjeti" da tekstualna datoteka s ograničavajućim kutijama već postoji za tu sliku, te će ih učitati i prikazati.

Program `label_characters` prilikom pokretanja očekuje putanju do slike retka teksta i putanju do transkripta označenih redaka stranice kojoj redak teksta pripada kao argumente naredbenog retka. Program zna koji red u transkriptu mora pročitati pomoću naziva slike retka teksta. Program će unaprijed za nas/automatski prilikom

[3]Označuje se držanjem gumba Ctrl, držanjem lijevog gumba miša i micanjem miša.

[4]Nazivi slika idu od 0 do $n-1$ gdje je n broj označenih redaka teksta.



Slika 3.2: Označeni znakovi

pokretanja potražiti znakove kao spojene komponente retka teksta (povezane komponente koje su jedna iznad druge, poput komponeti slova "i", grupa u jednu zajedničku komponentu) i označiti ih pravokutnicima a iznad pravokutnika i (pravokutnici sortirani po x osi) će stajati i-ti znak retka teksta pročitanog iz transkript datoteke (slika 3.2).

Program `label_characters` uz označavanje pravokutnikom i mogućnost ispravljanja pravokutnika što nudi i program `label_text_lines`, dodatno nudi cijepanje/rezanje pravokutnika vertikalnim pravcem[5]. Nakon što smo završili s označavanjem redaka zatvaranjem programa spremaju se ograničavajuće kutije (engl. bounding box) označenih znakova u tekstualnu datoteku istim imenom kao i slika retka teksta s kojom smo pokrenuli program, stvara se direktorij s istim imenom u koji se stvaraju slike znakova[6] i datoteka labels.txt u kojoj su zapisani parovi (naziv slike, labela) u svakom redu zasebno. Ako ponovno pokrenemo program s istom slikom, program će "vidjeti" da tekstualna datoteka s ograničavajućim kutijama već postoji za tu sliku, te će ih učitati i prikazati. S kombinacijom Crtl + R program ponovno učitava transkript datoteku što može biti korisno u slučaju da nam je prijepis kriv da ne moramo ponovno pokretati program.

3.3. Vrednovanje razvijenog sustava

Kod vrednovanja sustava gledali smo greške prepoznavanja tekst i brzinu izvođenja. Za greške prepoznavanja teksta koristiti smo Levenshteinovu udaljenost. Levenshte-

[5]Držanjem gumba Shift vertikalni pravac će se pojaviti/prikazati na x poziciji miša. Vertikalni pravac upravljamo mišem. Klikom na lijevi gumb miša izvršavamo cijepanje/rezanje.

[6]Nazivi slika idu od 0 do $n-1$ gdje je n broj označenih znakova.

Since the orientation is perfectly horizontal, the first step is to determine the position and extent of the lines of text in the image. This can be done by constructing a horizontal *projection* and searching it for minima. The projection is simply the sum of the pixel values in a specified direction, so a horizontal projection is the sum of the pixels in each row. The row of pixels that begins a new line will be one in which some of the pixels are black, and the last row belonging to that line will be the last one having any black pixels. The start and end columns for the line are found by searching the rows that belong to that line, from column zero through to the first column having a set pixel. The same is done, but in the reverse direction, to find the last set pixel in a line.

Slika 3.3: Slika skeniranog teksta

inova udaljenost je metrika niza za mjerenje razlike između dva niza. Neformalno, Levenshteinova udaljenost između dvije riječi najmanji je broj uređivanja od jednog znaka (umetanja, brisanja ili zamjene) potrebnih za promjenu jedne riječi u drugu.

Na primjer, Levenshteinova udaljenost između riječi "kitten" i "sitting" je 3, jer se sljedeća tri uređivanja mijenjaju jedno u drugo:

1. kitten $\rightarrow$ sitten (zamjena "s" za "k")
2. sitten $\rightarrow$ sittin (zamjena "i" za "e")
3. sittin $\rightarrow$ sitting (umetanje "g" na kraju).

Levenshteinovu udaljenost možemo gledati kao potreban broj ispravki što je motivirano ručnom korekcijom koju bi trebali napraviti da prepoznati tekst ispravimo u stvarni.

U nastavku prikazujemo kako sustav radi na dva primjera. Testirani sustav je OCR dan u 3.1.5.

3.3.1. Prepoznavanje skeniranog engleskog teksta

Testirali smo sustav sa slikom teksta 3.3[7]. Izgradnja skupa za učenje je objašnjena u pododjeljku 2.3.4 (Pružanje predložaka za svaki znak). Predloške za učenje smo normalizirali graničnim okvirom dimenzija 32×32.

Koristili smo `HorizontalProjection` za segmentaciju redaka, `ConcompSegmentationWithCharRecognition` za prepoznavanje redaka i `CharFeatureExt` za izvlačenje značajki znaka retka teksta (koristili smo piksele kao značajke). Za klasifikaciju smo koristili 1NN (nearest neighbor).

[7] Slika je uzeta sa web stranice www.wiley.com/go/jrparker.

Rezultat prepoznavanja:

```
Since the orienmtion iS perfect1y horiz0nml, the hrSt Step iS t0 dete%ine the wSiti0n and extent of me
lineS of text in the image- TbiS can be done by conSAcdng a h0hz0nG prdJccr/p4 1d Se|chLg it
f0r minima.  The pr0jecti0n iS Simply me Suln 0f the pixeI vMueS in a Sp%ihM dirxdon, So a ho+0n.
G projecti0n iS the Sum 0f the pixc1S in Och row.  The r0w 0f pixc1S that beginS a new 1ine wil1 M one
in which Some of me pi- xeIS ye bIack, zd the laSt r0w beIonging t0 that line will M me laSt one hav-
ing any black pixelS. Thc S0d aud end c01umuS f0r the line ye f0und by se1ching the rowS mat
belong to mat 1ine, from coIumn zcr0 thr0ugh to the hrSt c01umn having a Set pixel.  The Sue iS done,
but in me reVerSe dLxtion, t0 hnd mc laSt Set pixel in a 1ine.
```

Levenshteinova udaljenost između prepoznatog teksta i stvarnog je 183. U rezultatu vidimo da sustav zamjenjuje mala slova s velikim istog izgleda ("s" sa "S"). Ako oba teksta (prepoznati i stvarni) prebacimo u mala slova Levenshteinova udaljenost ispadne 146. Ako umjesto piksela kao značajki koristimo GSC-značajke Levenshteinova udaljenost ispadne 133 (prepoznati i stvarni tekst prebačeni u mala slova). Ako u skupu za učenje ostavimo samo znakove koji se pojavljuju na slici, Levenshteinova udaljenost ispadne 91.

Rezultat prepoznavanja:

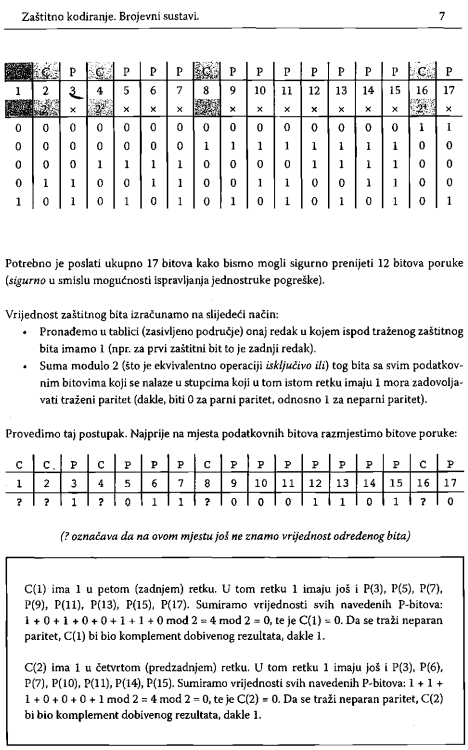
```
since the orienution is perfectly horizonul, the arst step is to detemine the fsition and extent of he lines of text
in the image- this can be done by consmcdng a hodzond pfojfcffon nd sehchhg it for minima- the projection is simply
he suin of the pixei vdues in a sphiad dirkdon, so a hohon.  m projection is the sum of the pixels in uch row.  the
row of pixels that begins a new line will h one in which some of he pi- xels he black, ud the last row belonging to
that line wiil h ae last one hav- ing any black pixels.  the suh and end columns for the line he found by sekching the
rows hat belong to hat line, from column zero through to the arst column having a set pixel.  the sme is done, but in
he reverse dkhtion, to and he last set pixel in a line-
```

Primjetimo da se na većini mjesta "." prepoznaje kao "-". To je zbog normalizacije graničnim okvirom - oba znaka, "." i "-", nakon normalizacije su više-manje crna slika. Time bi mogli izbjeći ako u skupu značajki dodamo omjer visine i širine znaka (prije normalizacije).

Probali smo koristiti segmentaciju temeljenu za prepoznavanju, ali ne daje dobre rezultate. Pretpostavljamo da je to zbog premalo uzoraka/podataka za treniranje.

Što se tiče brzine izvođenja, OCRu treba 2-4 sekunde[8] za se izvrši za sliku 3.3.

[8]Koristili smo računalo s procesorom Intel i5-3320M 2.60GHz.



Zaštitno kodiranje. Brojevni sustavi. 7

C	C	P	C	P	P	P	C	P	P	P	P	P	P	P	C	P
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
		×		×	×	×		×	×	×	×	×	×	×		×
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Potrebno je poslati ukupno 17 bitova kako bismo mogli sigurno prenijeti 12 bitova poruke (*sigurno* u smislu mogućnosti ispravljanja jednostruke pogreške).

Vrijednost zaštitnog bita izračunamo na slijedeći način:

- Pronađemo u tablici (zasivljeno područje) onaj redak u kojem ispod traženog zaštitnog bita imamo 1 (npr. za prvi zaštitni bit to je zadnji redak).
- Suma modulo 2 (što je ekvivalentno operaciji *isključivo ili*) tog bita sa svim podatkovnim bitovima koji se nalaze u stupcima koji u tom istom retku imaju 1 mora zadovoljavati traženi paritet (dakle, biti 0 za parni paritet, odnosno 1 za neparni paritet).

Provedimo taj postupak. Najprije na mjesta podatkovnih bitova razmjestimo bitove poruke:

C	C	P	C	P	P	P	C	P	P	P	P	P	P	P	C	P
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
?	?	1	?	0	1	1	?	0	0	0	1	1	0	1	?	0

(? označava da na ovom mjestu još ne znamo vrijednost određenog bita)

C(1) ima 1 u petom (zadnjem) retku. U tom retku 1 imaju još i P(3), P(5), P(7), P(9), P(11), P(13), P(15), P(17). Sumiramo vrijednosti svih navedenih P-bitova: 1 + 0 + 1 + 0 + 0 + 1 + 1 + 0 mod 2 = 4 mod 2 = 0, te je C(1) = 0. Da se traži neparan paritet, C(1) bi bio komplement dobivenog rezultata, dakle 1.

C(2) ima 1 u četvrtom (predzadnjem) retku. U tom retku 1 imaju još i P(3), P(6), P(7), P(10), P(11), P(14), P(15). Sumiramo vrijednosti svih navedenih P-bitova: 1 + 1 + 1 + 0 + 0 + 0 + 1 mod 2 = 4 mod 2 = 0, te je C(2) = 0. Da se traži neparan paritet, C(2) bi bio komplement dobivenog rezultata, dakle 1.

Slika 3.4: Skenirana stranica knjige

3.3.2. Prepoznavanje skenirane stranice knjige na hrvatskom jeziku

Koristeci alat za označavanje koji smo opisali u (3.2) označili smo 6 skeniranih stranica knjige na hrvatskom jeziku. Označavanjem je dobiveno 5752 slika znakova za učenje.

Testirali smo sustav sa skeniranom stranicom iste knjige 3.4 koju nismo označavali alatom. Slike znakova za učenje smo normalizirali graničnim okvirom dimenzija 32×32.

Koristili smo `RunLengthSmearing` za segmentaciju redaka, `ConcompSegmentationWithCh` za prepoznavanje redaka i `CharFeatureExtractor` za izvlačenje značajki znaka retka teksta (koristili smo piksele kao značajke). Za klasifikaciju smo koristili kNN ($k = 5$).

Rezultat prepoznavanja:

```
Zaštitno kodiranje.  Brojevni sustavi.

potrebno je poslati ukupno 12 bitova kako bismo mogli sigurno prenijeti 12 bitova poruke

(sknrno u smislu mogućnosti ispravljanja jednostruke pogreške).

vrijednost zaštitnog bita izračunamo na slijedeći način:

Pronademo u tablici (zasivljeno područje) onaj redak u kojem ispod traženog zašutnog
```

```
bita imamo 1 (npr.  za pni zaštitni bit to je zadnji redal().

suma modulo 2 (što je ekvivalenmo operaciji bi/jkćjvo i/i) tog bita sa svim podatkov-

nim bitovima koji se nalaze u stupcima koji u tom istom retku imaju 1 mora zadovolja.

vati traženi paritet (dakle, biti 0 za parni paritet, odnosno 1 za neparni paritet).

Provedimo taj postupak.  Najprije na mjeta podatkovnih bitova razmjestimo bitove poruke:

r.p ozndbvn dn nd ovom  jestn još ne znnn,o vrp.ednost odredenot biruj

c(1) ima 1 u petom (zadnjem) re+u.  U tom retku 1 imaju još i P(3), P(5), p(2),

P(9), P(11), p(13), P(15), P(1?).  Sumiramo vrijednosti svih navedenih P-bitova:

l + 0 + 1 + o + 0 + 1 + 1 + o mod 2 = 4 mod 2 = o, te je c(1) = o.  Da se traži neparan

paritet, c(1) bi bio komplement dobivenog rezultata, dakle 1.

c(2) ima 1 u čeUrtom (predzadnjem) retku.  U tom retku 1 imaju još i P(3), P(6),

P(Tj, P(10), P(1 1j, P(14), p(15j.  sriramo vrijednosti svih navedenih P-bitova:  1 + 1 +

1 + 0 + o + 0 + 1 mod 2 :  4 mod 2 = 0, te je c(2) = 0.  Da se traži neparan paritet, c(2)

bi bio komplement dobivenog rezultata, dakle 1.
```

Levenshteinova udaljenost između prepoznatog teksta i stvarnog je 80.

Pošto za ovaj primjer imamo više podataka odlučili smo isprobati neuronsku mrežu. Ostavili smo sve defaultne vrijednosti razreda `MLPClassifier` paketa scikit-learn (verzija 0.24.2)[9]. U tom slučaju je Levenshteinova udaljenost između prepoznatog teksta i stvarnog 68.

Probali smo koristiti segmentaciju temeljenu za prepoznavanju, ali i u ovom slučaju daje lošije rezultate.

[9]Defaulni parametri:

```
hidden_layer_sizes=100, activation='relu', *, solver='adam',
alpha=0.0001, batch_size='auto', learning_rate='constant',
learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True,
random_state=None, tol=0.0001, verbose=False, warm_start=False,
momentum=0.9, nesterovs_momentum=True, early_stopping=False,
validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08,
n_iter_no_change=10, max_fun=15000
```

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

4. Zaključak

Rad se bavio samo tiskanim tekstom dobro očuvanih skeniranih dokumenata. Navedeni su osnovni postupci za prepoznavanje tiskanog teksta sa slike skeniranog dokumenta. Razvijen je jednostavan prototip kao dokaz koncepta. Razvijen je i alat za izradu skupa podataka za treniranje modela za prepoznavanje znakova sa slike.

Nažalost dobiveni produkt nije upotrebljiv u stvarnosti jer je trenutno učen samo na jednom tipu fonta. Jedan od najvećih problema trenutnog rješenja je ne mogućnost segmentiranja znakova kad su spojeni ili rastavljeni. Implementirali smo pristupe za rukovanje tih problema (segmentacija temeljena na prepoznavanju), ali su se rezultati pokazali lošijim od jednostavnijih pristupa (koji pretpostavljaju da takvi slučajevi/problemi ne postoje).

Daljnji koraci su detaljnije testiranje i evaluacija implementiranih algoritama te proučavanje i implementacija naprednijih pristupa, poput skrivenog Markovljevog modela (engl. hidden Markov model) kojim se izbjegava "ručna" segmentacija znakova. Treba testirati kako bi postojeće rješenje funkcioniralo s različitim fontovima i različitim tipovima dokumenata kako bi sustav bio više upotrebljiv.

LITERATURA

Thomas Breuel. Character recognition, 2010. URL `https://sites.google.com/a/iupr.com/dia-course/lectures/lecture-07-character-recognition`. [Online; accessed 27-May-2021].

T.M. Breuel. Segmentation of handprinted letter strings using a dynamic programming algorithm. U *Proceedings of Sixth International Conference on Document Analysis and Recognition*, stranice 821–826, 2001. doi: 10.1109/ICDAR.2001.953902.

C.J.C. Burges, O. Matan, Y. Le Cun, J.S. Denker, L.D. Jackel, C.E. Stenard, C.R. Nohl, i J.I. Ben. Shortest path segmentation: a method for training a neural network to recognize character strings. U *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, svezak 3, stranice 165–172 vol.3, 1992. doi: 10.1109/IJCNN.1992.227175.

R.G. Casey i E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7): 690–706, 1996. doi: 10.1109/34.506792.

David Doermann i Karl Tombre. *Handbook of Document Image Processing and Recognition*. Springer Publishing Company, Incorporated, 2014. ISBN 0857298585.

John T. Favata i Geetha Srikantan. A multiple feature/resolution approach to handprinted digit and character recognition. *International Journal of Imaging Systems and Technology*, 7(4):304–311, 1996. doi: https://doi.org/10.1002/(SICI)1098-1098(199624)7:4<304::AID-IMA5>3.0.CO;2-C.

Jim R Parker. *Algorithms for image processing and computer vision*. John Wiley & Sons, 2010.

Danijel Radošević. Postupci i problemi optičkog prepoznavanja teksta. *Journal of Information and Organizational Sciences*, 20:17–31, 1996.

Rocchini. File:minimum bounding rectangle.svg — Wikipedia, the free encyclopedia, 2012. URL https://en.wikipedia.org/wiki/Minimum_bounding_rectangle#/media/File:Minimum_bounding_rectangle.svg. [Online; accessed 25-June-2021].

Faisal Shafait. Lecture 06 - document image analysis, 2011. URL https://sites.google.com/a/iupr.com/dia-course/lectures/lecture-document-image-analysis. [Online; accessed 27-May-2021].

Faisal Shafait i Thomas M. Breuel. A simple and effective approach for border noise removal from document images. U *2009 IEEE 13th International Multitopic Conference*, stranice 1–5, 2009. doi: 10.1109/INMIC.2009.5383115.

Wikipedia contributors. Minimum bounding rectangle — Wikipedia, the free encyclopedia, 2020. URL https://en.wikipedia.org/w/index.php?title=Minimum_bounding_rectangle&oldid=968512545. [Online; accessed 25-June-2021].

K. Y. Wong, R. G. Casey, i F. M. Wahl. Document analysis system. *IBM Journal of Research and Development*, 26(6):647–656, 1982. doi: 10.1147/rd.266.0647.

P. Yin. Skew detection and block classification of printed documents. *Image Vis. Comput.*, 19:567–579, 2001.

Automatsko prepoznavanje tiskanog teksta

Sažetak

Računalno prepoznavanje teksta sa slike (primjerice dobivene skeniranjem dokumenta) težak je problem koji je moguće rješavati različitim pristupima. Tekst pri tome može biti rukom pisan, ili pak tiskan. Najjednostavniji se pristupi temelje na nizu slijednih koraka poput segmentacije pojedinih slova, klasifikacije segmentiranog slova, grupiranja segmentiranih slova u riječi i slično. U ovom radu objašnjavamo osnovne postupke za prepoznavanje tiskanog teksta, pri čemu se ograničavamo na slučaj kada su riječi pisane hrvatskim i engleskim jezikom. Razvijena je prototipna programska implementacija sustava koji na ulazu dobiva sliku odskenirane stranice teksta (primjerice, stranicu knjige), a na izlazu daje prepoznati tekst. U radu je napravljeno i vrednovanje razvijenog sustava.

Ključne riječi: Optičko prepoznavanje znakova, Obrada slike dokumenta, Prepoznavanje slike dokumenta

Automatic recognition of printed text

Abstract

Computer recognition of text from an image (for example, obtained by scanning a document) is a difficult problem that can be solved by different approaches. The text can be handwritten or printed. The simplest approaches are based on a series of sequential steps such as segmentation of individual letters, classification of segmented letters, grouping of segmented letters into words, etc. In this paper, we explain the basic procedures for recognizing printed text, limiting ourselves to the case when words are written in Croatian and English. A prototype software implementation of the system has been developed, which receives an image of a scanned page of text (for example, a book page) at the input, and provides recognized text at the output. The paper also evaluates the developed system.

Keywords: Optical character recognition, Document image proccessing, Document image recognition