

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5709

**Sustav za određivanje strukture
teksta na temelju položaja
pojedinih znakova**

Herman Zvonimir Došilović

Zagreb, lipanj 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 14. ožujka 2018.

ZAVRŠNI ZADATAK br. 5709

Pristupnik: Herman Zvonimir Došilović (0036480275)
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: Sustav za određivanje strukture teksta na temelju položaja pojedinih znakova

Opis zadatka:

Sustavi za automatsko očitavanje teksta sa skeniranih dokumenata imaju nekoliko zadaća koje uključuju lokalizaciju, segmentaciju i prepoznavanje pojedinih znakova te slaganje prepoznatih znakova u složenije strukture poput riječi i linija. To je u praksi vrlo težak problem.

U okviru ovog završnog rada potrebno je proučiti načine za određivanje riječi i linija na temelju položaja individualnih znakova te njihovih omeđujućih pravokutnika. U okviru rada potrebno je pripremiti odgovarajući skup podataka za testiranje te napraviti prototipnu implementaciju sustava.

Radu priložiti izvorni kod razvijenih postupaka uz potrebna objašnjenja i dokumentaciju. Predložiti pravce budućeg razvoja. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 16. ožujka 2018.

Rok za predaju rada: 15. lipnja 2018.

Mentor:

Doc. dr. sc. Marko Čupić

Predsjednik odbora za
završni rad modula:

Prof. dr. sc. Siniša Srblijić

Djelovodja:

Doc. dr. sc. Tomislav Hrkać

Zahvaljujem svom mentoru doc. dr. sc. Marku Čupiću na dozvoli za odabir vlastite teme i na strpljenju, poticaju i savjetima u razvoju rada.

Zahvaljujem tvrtki Microblink na danim sredstvima bez kojih ovaj rad ne bi bio moguć. Posebno zahvaljujem kolegama Jurici Cerovecu, Nenadu Mikši, Borisu Trubiću, Igoru Smolkoviču i Ivanu Jurinu koji su me svojim bogatim znanjem i iskustvom usmjeravali u razvoju rada.

Tko hoće da među vama bude najveći, neka vam bude poslužitelj! I tko hoće da među vama bude prvi, neka bude svima sluga. - Mk 10,43-44

SADRŽAJ

1. Uvod	1
2. Optičko raspoznavanje znakova	2
2.1. Primjene	2
2.2. Komponente OCR-sustava	4
3. Određivanje strukture teksta	8
4. Određivanje strukture teksta na temelju položaja pojedinih znakova	13
4.1. Željena funkcionalnost	15
4.2. Skup podataka za testiranje	17
4.2.1. Slike	17
4.2.2. Ulazne datoteke	18
4.2.3. Očekivane izlazne datoteke	20
4.3. Korištenje skupa podataka za testiranje	21
5. Algoritmi za određivanje strukture teksta	22
5.1. Algoritmi za određivanje linija	22
5.1.1. Algoritam temeljen na maksimalnom preklapanju znakova . .	23
5.2. Algoritmi za rastavljanje riječi	28
5.2.1. Algoritam temeljen na prosječnoj širini znaka	29
6. Zaključak	31
Literatura	32

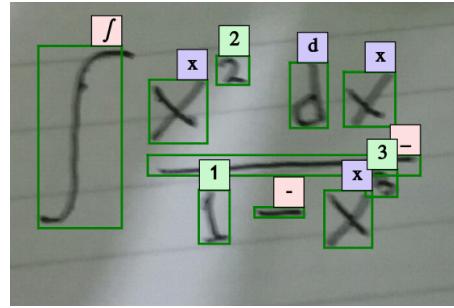
1. Uvod

2. Optičko raspoznavanje znakova

Sustav za optičko raspoznavanje znakova (engl. *optical character recognition*) (u daljnjem tekstu: *OCR-sustav*) pretvara sliku tiskanog teksta u digitalizirani format kojim možemo jednostavno manipulirati na računalu. Iako je to ljudima jednostavan zadatak, računalima nije lako prepoznati tekst i pojedine znakove teksta sa slike zbog velike raznolikosti jezika, fonta i stila kojim tekst može biti napisan. Optičko raspoznavanje znakova je stoga vrlo zahtjevan problem i mnogo je istraživačkog truda uloženo u pokušaju da se slike teksta pretvore u format koji računalo razumije. (Islam et al., 2017)

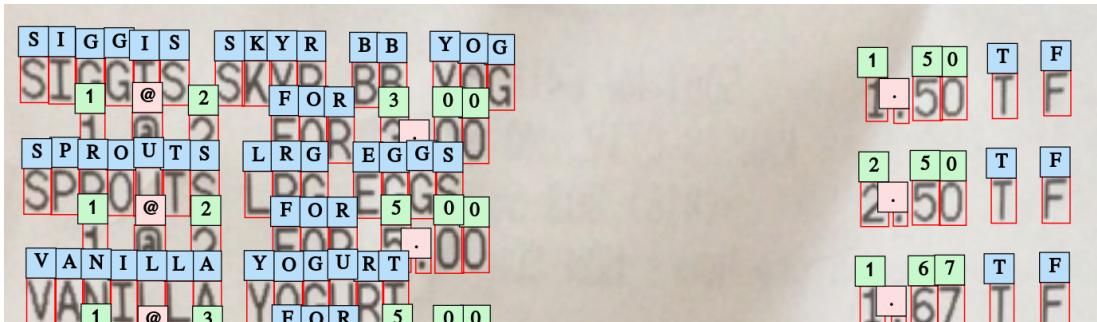
2.1. Primjene

Osim tiskanog teksta, OCR-sustavi se koriste i u prepoznavanju znakova rukom pisanih teksta. Prepoznavanje znakova rukom pisanih teksta je teži problem od prepoznavanja tiskanog teksta (Islam et al., 2017) zato jer se oblik znakova i njihov način pisanja razlikuje kod svake osobe (npr. rukopis odrasle osobe potpuno je drugačiji od rukopisa djeteta). OCR-sustave za detekciju rukom pisanih znakova možemo podijeliti na dvije potkategorije: *on-line* i *off-line*. *On-line* OCR-sustavi detektiraju znakove dok ih korisnici unose i to im omogućuje praćenje parametara poput: brzine pisanja, broja napravljenih poteza, smjer pisanja, itd. *Off-line* OCR-sustavi izvode se nad jednom slikom na kojoj se nalazi sav sadržaj nad kojim je potrebno napraviti detekciju. Takvi sustavi nemaju dodatne informacije koje imaju *on-line* sustavi i zato je detekcija znakova komplikiranija (Islam et al., 2017). Slika 2.1 prikazuje primjer rezultata *off-line* OCR-sustava za detekciju rukom pisanih znakova.

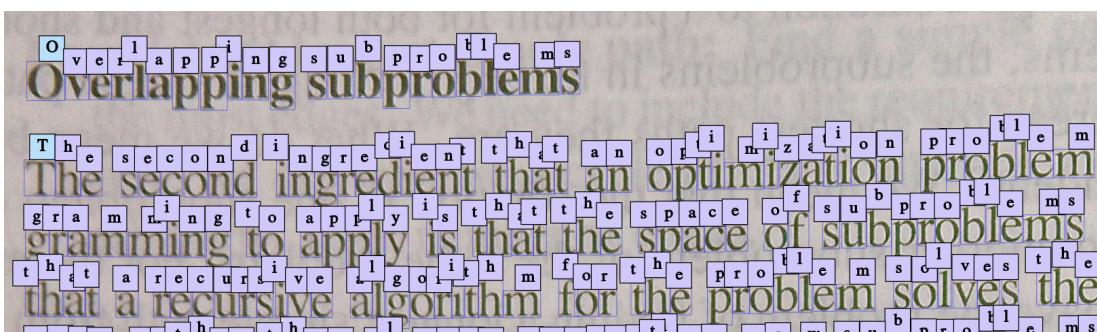


Slika 2.1: Rezultat *off-line* OCR-sustava za detekciju znakova rukom pisanog teksta.

OCR-sustavi imaju široku primjenu i možemo ih pronaći primjerice u detekciji znakova na registarskim pločicama (Saghaei, 2016), (Laroca et al., 2018), u detekciji znakova sadržaja knjiga (Wick et al., 2018), (Christy et al., 2017) i detekciji znakova na raznim dokumenatima (Harraj i Raissouni, 2015) (Verma et al., 2016). Na slici 2.2 prikazan je primjer rezultata korištenja OCR-sustava za detekciju znakova na računima iz trgovine. Slika 2.3 prikazuje rezultat OCR-sustava za detekciju znakova na tiskanim knjigama.



Slika 2.2: Rezultat OCR-sustava za detekciju znakova na računima iz trgovine.



Slika 2.3: Rezultat OCR-sustava za detekciju znakova na tiskanim knjigama.

2.2. Komponente OCR-sustava

Optičko raspoznavanje znakova provodi se u nekoliko koraka (Islam et al., 2017) (Kaur i Rani, 2016):

1. pribavljanje slike,
2. predobrada,
3. segmentacija znakova,
4. izdvajanje značajki znakova,
5. klasifikacija znakova i
6. naknadna obrada.

Pribavljanje slike

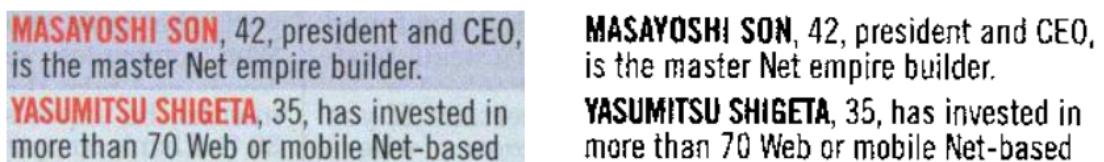
U prvom koraku OCR-a, pribavljanju slike, potrebno je pribaviti sliku nad kojom ćemo provesti ostale korake. Sliku možemo pribaviti s raznih uređaja poput kamere fotoaparata, mobilnog uređaja ili nekog drugog uređaja za digitalizaciju dokumenata (engl. *scanner*). Nakon prvog koraka, slika dokumenta nad kojim provodimo raspoznavanje znakova sastoji se samo od slikovnih elemenata (engl. *pixels*) (Vynckier, 2018). Slika 2.4 prikazuje primjer slike nad kojom možemo provesti postupak raspoznavanja znakova. Primjetimo da slika može sadržati pozadinu koju bi OCR-sustav trebao zanemariti.



Slika 2.4: Ulazna slika u OCR-sustav pribavljena kamerom mobilnog uređaja.

Predobrada

U predobradi slike OCR-sustavi često provode niz morfoloških transformacija i filtra nad pribavljenom slikom. Cilj ovog koraka je povećati kvalitetu slike i smanjiti informacije na slici. Binarizacija je jedan od potkoraka predobrade koji slike u boji ili u nijansama sive pretvara u crno-bijele. Osim binarizacije koriste se neke morfološke transformacije poput dilatacije, rezanja i skaliranja. Slika 2.5 prikazuje primjer slike prije i nakon binarizacije. (Gulan, 2016), (Islam et al., 2017), (Jurin, 2017)



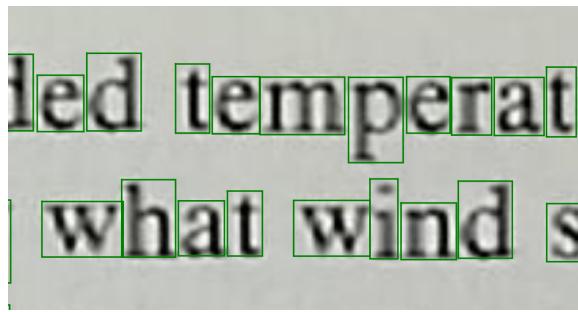
Slika 2.5: Prije binarizacije (lijevo) i nakon binarizacije (desno) (Vynckier, 2018).

Segmentacija znakova

Sljedeći korak, segmentacija znakova, je postupak segmentiranja slike u segmente unutar kojih se nalaze znakovi koje želimo klasificirati. Jedan od pristupa segmentacije izvodi se s vrha prema dnu gdje se prvo segmentiraju linije, zatim riječi i na kraju pojedini znakovi (Jurin, 2017), (Vynckier, 2018). Prednost ovakvog pristupa je da uz lokaciju svakog znaka dobivamo i strukturu cijelog teksta, odnosno, znamo kojoj liniji i kojoj riječi znak pripada. Nedostatak ovakvog pristupa je da ne postoje korekcijski mehanizmi kojima bismo znak pridružili nekoj drugoj liniji ili riječi ako su prva dva koraka segmentacije linije ili riječi neispravni. (Jurin, 2017)

Drugi pristupi poput *ZICER OCR*¹ sustava izravno izvode segmentaciju cijele slike na području koji predstavljaju znakove. Prednost takvog pristupa je da možemo detektirati znakove teksta u kojemu nema riječi i linija, kao što je na primjer matematički izraz. Nedostatak takvog pristupa je da gubimo informaciju o strukturi teksta i zato postoji potreba za razvojem dodatnog sustava koji bi znakove grupirao u riječi, a riječi u linije (Jurin, 2017). Slika 2.6 prikazuje rezultat segmentacije pojedinih znakova.

¹OCR-sustav tvrtke *Microblink*, <https://microblink.com>



Slika 2.6: Segmentacija znakova.

Izdvajanje značajki

Izdvajanje značajki pojedinog znaka podrazumijeva odabir značajki prema kojima će se jedinstveno klasificirati svaki znak. Značajke poput geometrijskog oblika ili statističkih svojstava mogu biti uzete u obzir prilikom klasifikacije. Važno područje istraživanja pripada razmatranju koje i koliko značajki je potrebno uzeti u obzir za kvalitetnu i ispravnu klasifikaciju. (Islam et al., 2017)

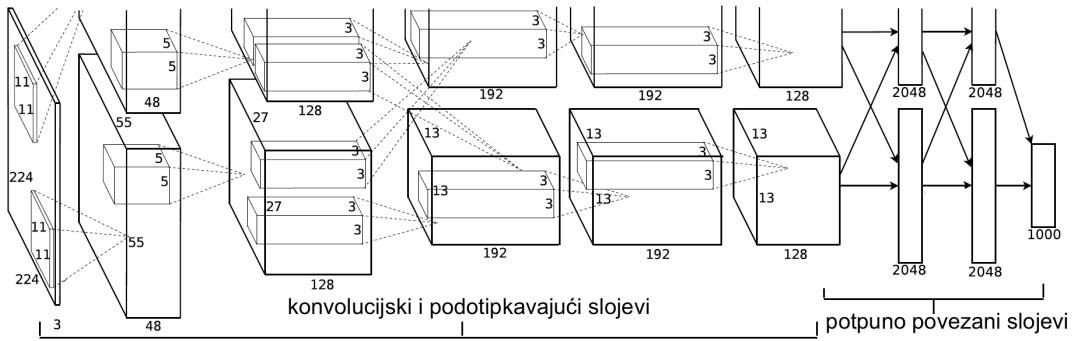
Klasifikacija

Klasifikacija je najvažniji korak optičkog raspoznavanja znakova (Verma i Ali, 2012) (Zhu et al., 2016) koji koristi izdvojene značajke za određivanje klase pojedinog znaka (Lehal i Singh, 1999) (Kaur i Rani, 2016). Statistički pristupi klasifikacije koriste diskriminativne funkcije za određivanje klase znaka (Islam et al., 2017), a u novije vrijeme koriste se duboke neuronske mreže (Jurin, 2017). Neki od statističkih pristupa su: Bayesov klasifikator, klasifikator stablom odluke, umjetne neuronske mreže i metoda k-najbližih susjeda (Islam et al., 2017).

2012. godine Krizhevsky i suradnici (Krizhevsky et al., 2012) objavili su rad koji je označio prekretnicu u klasifikaciji i lokalizaciji objekata (Jurin, 2017). Slika 2.7 prikazuje arhitekturu *AlexNet* koja je pobijedila na natječaju *ImageNet 2012* u području klasifikacije objekata. (Jurin, 2017)

Naknadna obrada

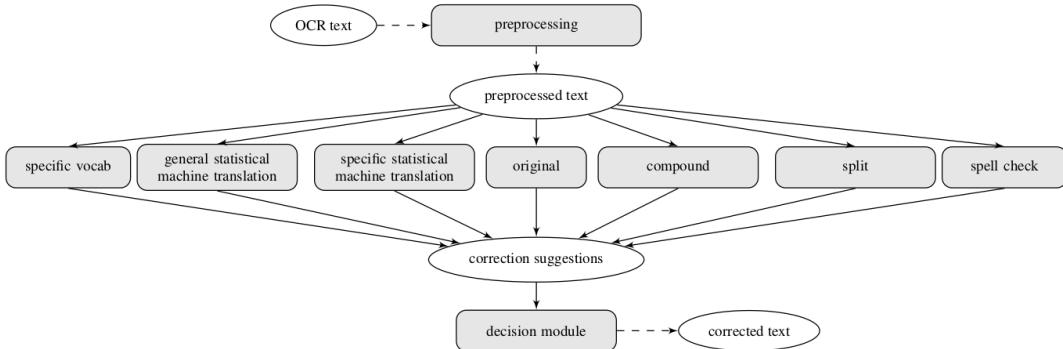
Nakon klasifikacije znakova slijedi njihova naknadna obrada koja se koristi kako bi se poboljšali OCR-rezultati. Jedan od pristupa postprocesiranja koristi rezultate više različitih klasifikatora koji mogu biti korišteni slijedno, paralelno ili hijerarhijski. Nakon toga rezultati klasifikatora se kombiniraju različitim pristupima. (Islam et al., 2017) Kao što je spomenuto u 2.2 segmentacija koja se ne provodi s vrha prema dnu nema



Slika 2.7: Arhitektura *AlexNet* (Jurin, 2017)

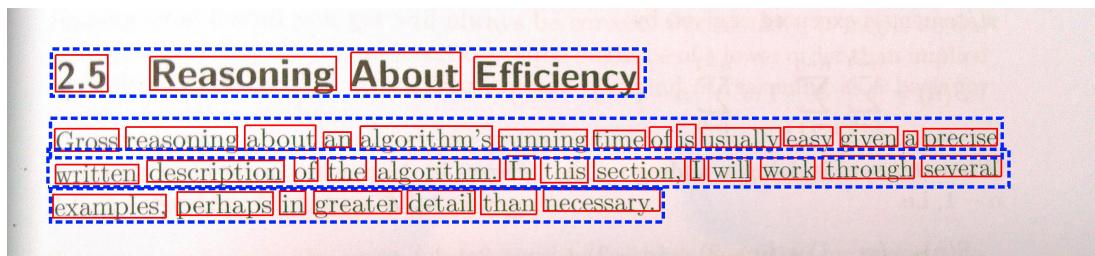
informaciju o strukturi teksta i zato je potrebno razviti dodatan **sustav za određivanje strukture teksta na temelju položaja pojedinih znakova**.

Schulz i suradnici (Schulz i Kuhn, 2017) 2017. godine predstavili su arhitekturu tzv. *post-correction* OCR-sustava kojim su pokazati na koji način su adaptirali generički sustav za naknadnu OCR-rezultata koristeći domensko znanje za konkretni problem koji su rješavali. Ovim pristupom ostvarili su bolje rezultate za konkretni problem nego što su ostvarili koristeći postojeći generički sustav za naknadnu obradu OCR-rezultata.



Slika 2.8: Arhitektura *post-correction* OCR sustava (Schulz i Kuhn, 2017)

3. Određivanje strukture teksta



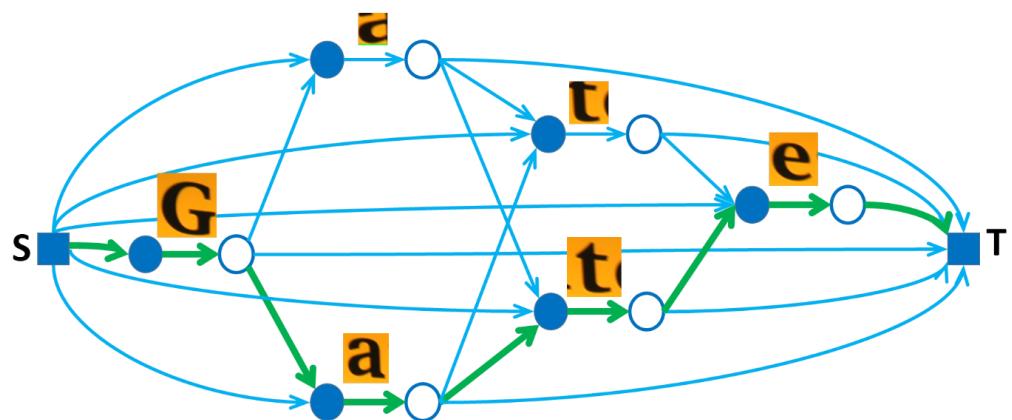
Slika 3.1: Segmentacija linija (plavo) i riječi (crveno) unutar linije.

Sustavi za određivanje strukture teksta na temelju OCR-rezultata sastavni su dio OCR-sustava. Određivanje strukture teksta podrazumjeva segmentaciju linija i segmentaciju riječi unutar linije. Neke tehnike segmentacije znakova i njihove klasifikacije nemaju informaciju o tome kojoj liniji i riječi pojedini znak pripada. Prednost takvog pristupa je da takav OCR-sustav možemo koristiti nad slikama koje ne sadrže linije, kao što su na primjer slike matematičkih izraza (Jurin, 2017). Nedostatak je što nakon klasifikacije moramo razviti sustav koji će znakove naknadno obraditi da bismo odredili strukturu teksta (Jurin, 2017).

Tekst na slici može biti podijeljen na linije ili blokove, a u bloku tekst možemo podijeliti na linije. Unutar jedne linije znakove možemo grupirati riječi. Način na koji će se odrediti struktura teksta uvelike ovisi o problemu koji riješavamo i kakve rezultate želimo dobiti. Tekst na slici 3.1 podijeljen je na linije (plavo), a unutar svake linije na riječi (crveno).

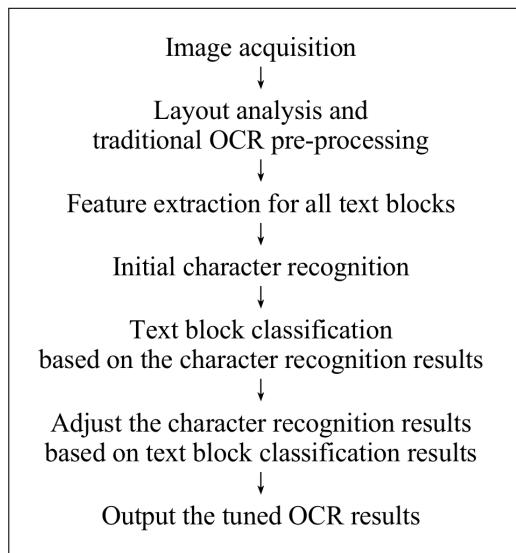
Tain i suradnici (Tian et al., 2016) predlažu sustav za određivanje strukture teksta koji će osim određivanja kojoj liniji pojedini znak pripada znati izbaciti tzv. *false positive* znakove odnosno znakove koje je OCR-sustav prepoznao, a zapravo u tekstu ne postoje. Njihov sustav temelji se na *min-cost flow network* modelu koji objedinjuje izbacivanje *false positive* znakova i pronalazak strukture teksta. Na temelju međusobne pozicije između dva prepoznata znaka i dodatnog parametra kojeg dobivaju od klasifikatora, a koji označava vjerojatnost ispravne detekcije, grade težinski usmjereni graf

(Slika 3.2) kojim svoj problem modeliraju *min-cost flow network* modelom.



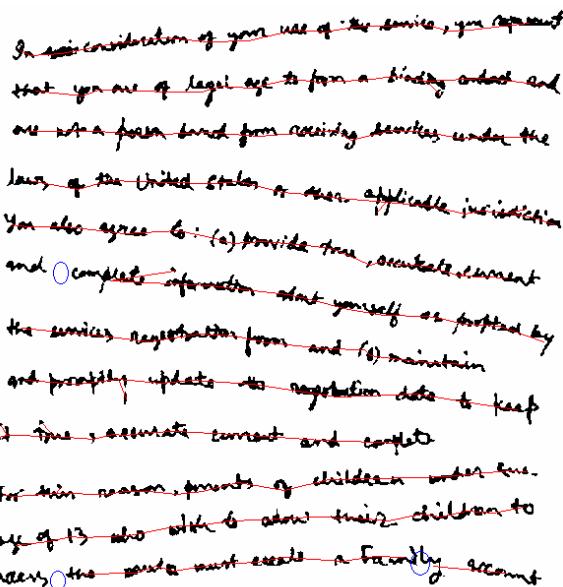
Slika 3.2: Težinski usmjereni graf temeljen na *min-cost flow network* modelu (Tian et al., 2016).

Zhu i suradnici (Zhu et al., 2016) predložili su novu arhitekturu (slika 3.3) OCR-sustava koji se temelji na empirijskim rezultatima koji su pokazali da sadržaj riječi ne ovisi samo o dijelu teksta u kojem se ta riječ nalazi nego i o susjednim dijelovima teksta. Njihov novi OCR-sustav radi dvostruku analizu strukture teksta – prije klasifikacije i nakon klasifikacije. Prva analiza strukture teksta omogućuje im da odrede strukturu teksta u blokovima, a druga analiza strukture teksta im omogućuje da poprave pogreške u klasifikaciji. Njihova nova arhitektura predstavlja hibridni OCR-sustav koji iskorištava rezultate analize strukture teksta.



Slika 3.3: Arhitektura novog OCR-sustava kojeg predlažu Zhu i suradnici (Zhu et al., 2016).

Yin i suradnici (Yin i Liu, 2007) pronalaze linije u tekstu povezujući znakove u težinski graf nad kojim provode Kruskalov algoritam za pronalazak minimalnog razapinjućeg stabla. Njihov pristup ne koristi rezultate klasifikacije, nego koriste povezane komponente koje im predstavljaju znakove i koje pronalaze koristeći algoritam temeljen na praćenju kontura (engl. *contour tracing*). Slika 3.4 prikazuje rezultat pronalaska linija teksta u rukom pisanim dokumentu na Engleskom jeziku.

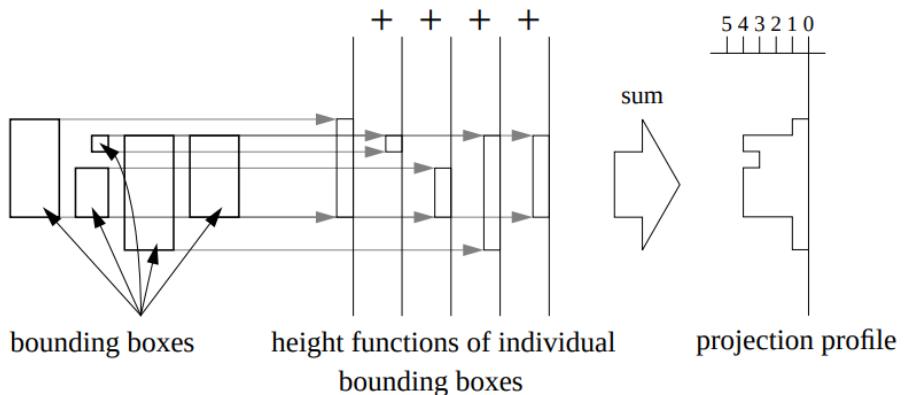


Slika 3.4: Rezultat pronalaska linija teksta u rukom pisanim dokumentu na Engleskom jeziku (Yin i Liu, 2007)

Motivirani njihovim radom Pan i suradnici (Pan et al., 2011) predstavili su sličan pristup koji u težinama grafa uzima u obzir dodatne težine koje su učene MCE (engl. *minimum classification error*) mjerom.

Još jedan pristup predložili su Yin i suradnici (Yin et al., 2013) koji koriste tehniku hijerahiskog grupiranja koji postupno spaja linije koje dijele znakove dok god postoje linije koje se mogu spojiti. (Tian et al., 2016)

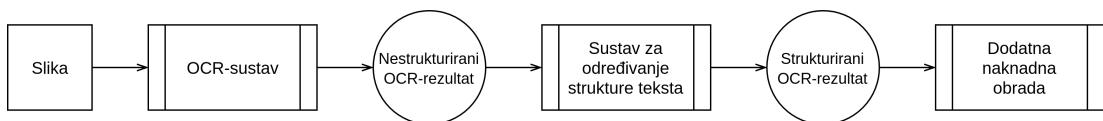
Liang i suradnici (Liang et al., 1996) predlažu heuristički algoritam za određivanje strukture teksta. Algoritam radi horizontalnu projekciju (slika 3.5) omeđujućih pravokutnika na jednu ravnicu i pronalazi vrhove i doline u histogramu koji prikazuje frekvencije pojavljivanja projektiranih pravokutnika. Osim ovog pristupa predložili su još jedan koji spaja dva znaka u jednu cjelinu ako i samo ako su dva znaka dovoljno blizu da ih ima smisla spojiti. Gupta i suradnici (Gupta et al., 2006) također koriste razne heuristike prema kojima povezuju susjedne omeđujuće pravokutnike.



Slika 3.5: Histogram dobiven horizontalnom projekcijom omeđujućih pravokutnika (Liang et al., 1996)

Određivanje strukture teksta težak je postupak koji uvelike ovisi o problemu koji rješavamo. U ovom poglavlju smo pokazali da strukturiranje teksta može biti izvođeno u raznim fazama izvođenja OCR-sustava. Trenutak u kojem ćemo pokrenuti analizu strukture teksta ovisi o načinu na koji smo označili segmente znakova, koje informacije o segmentu imamo i koji problem rješavamo. Pokazali smo da su neki pristupi postigli bolje rezultate kada se iskoristilo domensko znanje i kada su se u nekim heurističkim pristupima koristili parametri koji su bili pomno izabrani za dani problem. U nastavku ovog rada predložit ćemo nekoliko pristupa za određivanje strukture teksta na temelju položaja omeđujućih pravokutnika nakon provedene klasifikacije znakova. Analizirat ćemo njihove prednosti i nedostatke i cijelo vrijeme ćemo znati koji točno problem rješavamo.

4. Određivanje strukture teksta na temelju položaja pojedinih znakova



Slika 4.1: Suradnja OCR-sustava i sustava za određivanje strukture teksta.

U kontekstu određivanja strukture teksta na temelju položaja pojedinih znakova, radi preglednije analize, razmatramo OCR-sustav koji nema korak naknadne obrade. Nakon završetka klasifikacije OCR-sustav posjeduje nestrukturirani OCR-rezultat u kojemu se nalaze svi segmentirani i klasificirani znakovi. Takav nestrukturirani OCR-rezultat prosljeđuje se sustavu za određivanje strukture teksta na naknadnu obradu. U praksi je sustav za određivanje strukture teksta sastavni dio naknadne obrade OCR-sustava, međutim, u ovom završnom radu ćemo razdvojiti ta dva sustava radi preglednije analize njihove suradnje.

Sustav za određivanje strukture teksta na temelju položaja pojedinih znakova (u dalnjem tekstu: *Sustav*) na ulaz od OCR-sustava prima nestrukturirani OCR-rezultat koji u sebi sadrži sve znakove koje je OCR-sustav prepoznao. Za svaki znak u OCR-rezultatu znamo sljedeće informacije:

- x - horizontalnu poziciju gornjeg lijevog kuta,
- y - vertikalnu poziciju gornjeg lijevog kuta,
- $width$ - širinu znaka,
- $height$ - visinu znaka i
- $value$ - Unicode vrijednost znaka.

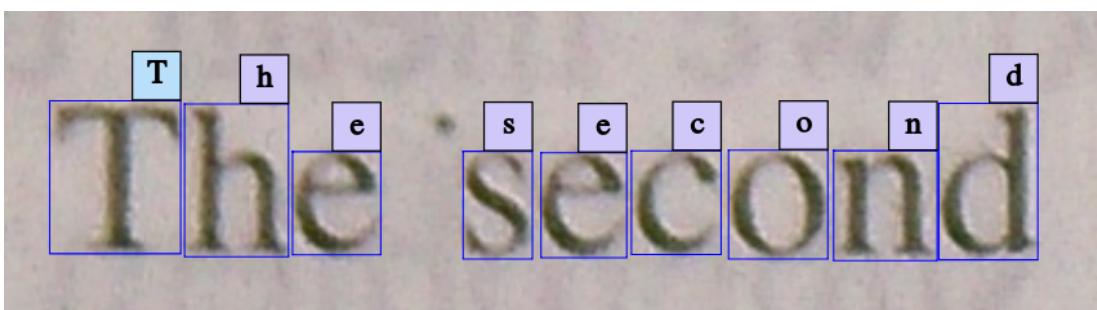
Izlaz sustava je strukturirani OCR-rezultat u kojemu su znakovi grupirani u linije, a unutar svake linije su riječi odvojene znakom bjeline. Strukturirani OCR-rezultat se

zatim prosljeđuje dodatnim naknadnim obradama koje ovise o problemu koji se rješava i u ovom završnom radu o njima neće biti riječi. Slika 4.1 prikazuje vizualizaciju navedenih suradnji.

U skolpu ovog završnog rada potrebno je razviti sustav za određivanje strukture teksta koji će rješavati problem određivanja strukture teksta na računima iz trgovine i sadržaja iz knjiga. Stup podataka za testiranje sustava biti će detaljno objašnjen u odjeljku 4.2.

Slika 4.2 prikazuje vizualizaciju rezultata OCR-sustava koji nam je za svaki znak vratio omeđujući pravokutnik (označeno plavom bojom) i vrijednost znaka odnosno klasu kojoj znak pripada. Slike 2.2 i 2.3 također prikazuju vizualizaciju rezultata OCR sustava i primjere s kakvim će se sustav za određivanje strukture teksta koji razvijamo susresti.

Primjer pojednostavljenog (detaljnije u odjeljku 4.2) nestrukturiranog OCR-rezultata u formatu JSON koji će sustav za određivanje strukture teksta dobiti kao svoj ulaz prikazan je isječkom 4.1. Ulazni nestrukturirani OCR-rezultat u formatu JSON uvijek se sastoji od jedne linije u koju su smješteni svi segmentirani i klasificirani znakovi u nasumičnim redoslijedom.



Slika 4.2: Vizualizacija OCR-rezultata.

Isječak 4.1: Pojednostavljeni nestrukturirani OCR-rezultat u formatu JSON.

```
1  {
2      "ocr_result": {
3          "lines": [
4              {
5                  "chars": [
6                      {
7                          "x": 25.95604, "y": 17.30562,
8                          "width": 10.64438, "height": 16.60289,
9                          "value": 48
10                     },
11                     {
12                         "x": 36.60289, "y": 17.30562,
13                         "width": 10.64438, "height": 16.60289,
14                         "value": 48
15                     }
16                 ]
17             }
18         ]
19     }
20 }
```

```

12         "x": 19.77133, "y": 1.28793,
13         "width": 16.07777, "height": 10.76925,
14         "value": 77
15     },
16     {
17         "x": 5.50248, "y": 2.84320,
18         "width": 12.13375, "height": 15.60966,
19         "value": 73
20     },
21     {
22         "x": 3.19550, "y": 19.67606,
23         "width": 14.94088, "height": 20.78798,
24         "value": 91
25     },
26
27     ]
28 }
29 ]
30 }
31 }
```

4.1. Željena funkcionalnost

Od sustava se očekuje da za dobiveni nestrukturirani OCR-rezultat u formatu JSON vrati novi strukturirani OCR-rezultat u istom formatu koji će znakove grupirati u linije i koji će unutar linija biti poredani s lijeva na desno. Također, linije moraju biti sortirane tako da se najviša linija u dokumentu nalazi na prvom mjestu.

Osim grupiranja linija od sustava se očekuje da između dva znaka, gdje smatra da završava prethodna i započinje nova riječ, ubaci novi znak bjeline čija je vrijednost (engl. *value*) 32 dok ostale informacije mogu biti proizvoljne. Dodatan zahtjev je da sustav sve grupirane linije smjesti u jedan blok. Detaljnije o blokovima će biti opisano u pododjeljku 4.2.2.

Isječak 4.2 prikazuje primjer izlaza iz sustava za dani ulaz iz isječka 4.1. Sustav je znakove grupirao u dvije linije i između prvog i zadnjeg znaka u drugoj liniji je ubacio znak bjeline. Vrijednost znaka bjeline je zahtijevana vrijednost 32. Ostale informacije znaka bjeline mogle su biti proizvoljne, međutim, sustav im je dodijelio sljedeće smislenije vrijednosti:

- x - horizontalna pozicija gornjeg desnog kuta lijevog znaka,
- y - vertikalna pozicija gornjeg lijevog kuta desnog znaka,

- *width* - horizontalna udaljenost između gornjeg desnog kuta lijevog znaka i gornjeg lijevog kuta desnog znaka,
- *height* - visina lijevog znaka.

Pod *lijevi znak* podrazumijeva se na znak koji se nalazi prije znaka bjeline, a pod *desni znak* podrazumijeva se na znak koji se nalazi nakon znaka bjeline.

Isječak 4.2: Izlaz iz sustava za određivanje strukture teksta u formatu JSON.

```

1  {
2      "ocr_result": {
3          "lines": [
4              {
5                  "chars": [
6                      {
7                          "x": 5.50248, "y": 2.84320,
8                          "width": 12.13375, "height": 15.60966,
9                          "value": 73
10                     },
11                     {
12                         "x": 19.77133, "y": 1.28793,
13                         "width": 16.07777, "height": 10.76925,
14                         "value": 77
15                     }
16                 ]
17             },
18             {
19                 "chars": [
20                     {
21                         "x": 3.19550, "y": 19.67606,
22                         "width": 14.94088, "height": 20.78798,
23                         "value": 91
24                     },
25                     {
26                         "x": 18.13638, "y": 19.67606,
27                         "width": 7.81966, "height": 20.78798,
28                         "value": 32
29                     }
30                 }
31             },
32             {
33                 "x": 25.95604, "y": 17.30562,
34                 "width": 10.64438, "height": 16.60289,
35                 "value": 48
36             }
37         ]
38     }

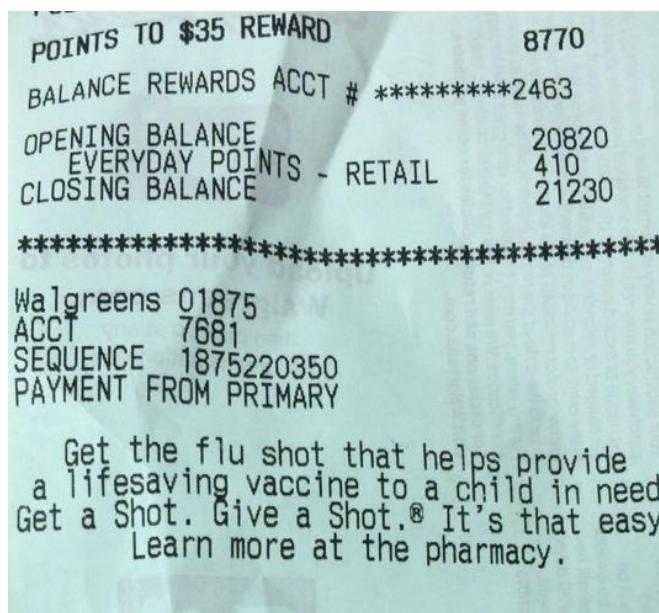
```

4.2. Skup podataka za testiranje

Skup podataka za testiranje (u dalnjem tekstu: *podatci*) sustava sastoji se od slika, ulaznih datoteka u formatu JSON i očekivanih izlaznih datoteka. U sklopu ovog završnog rada razvijeni sustav za određivanje strukture teksta rješavati će problem određivanja strukture teksta na računima iz trgovine i sadržaja iz knjiga.

4.2.1. Slike

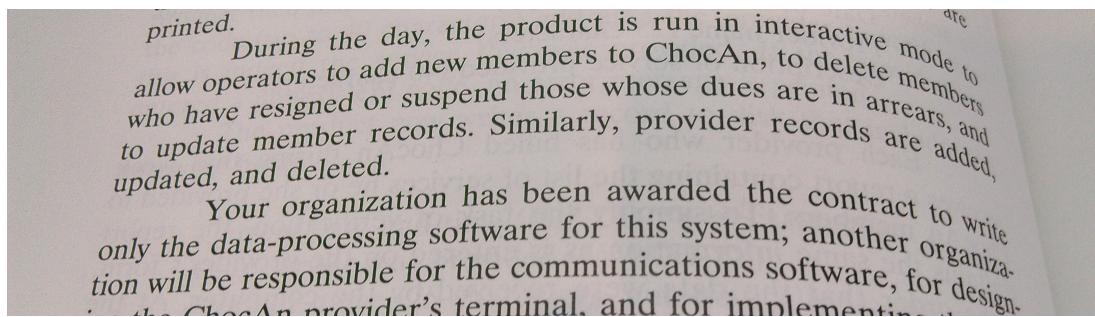
Podatci za testiranje sustava sadrže 100 slika računa (primjer na slici 4.3) i 34 slike sadržaja iz knjiga (primjer na slici 4.4).



Slika 4.3: Primjer slike računa iz trgovine.

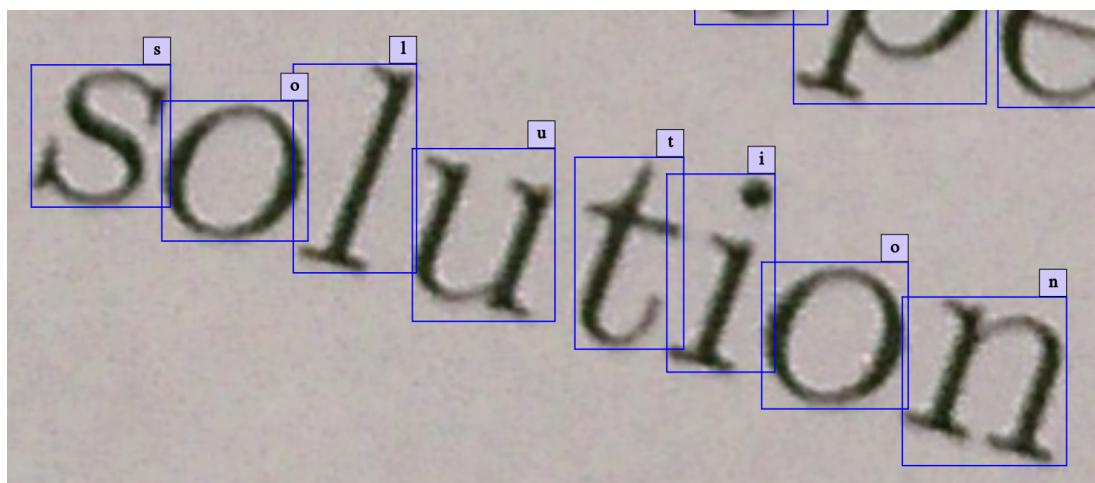
Svaki znak na svakoj slici je **ručno** označen i klasificiran. Na slikama računa iz trgovine označeno je ukupno 85068 znakova, a na slikama sadržaja iz knjiga označeno je ukupno 25092 znaka. Ručno označeni podatci oponašaju nestrukturirane OCR-rezultate koji su ulaz u sustav za određivanje strukture teksta.

Omeđujući pravokutnici označenih znakova predstavljaju područje koje označeni znak zauzima na slici. Stranice omeđujućih pravokutnika su uvijek paralelne sa rubovima slike. Slika 4.5 prikazuje isječak slike, sadržaja iz knjige, na kojoj su znakovi



Slika 4.4: Primjer slike sadržaja iz knjige.

ukošeni, a stranice njihovih omeđujućih znakova paralelne su sa rubovima slike. Možemo uočiti kako je moguće da se dva susjedna omeđujuća pravokutnika preklapaju.



Slika 4.5: Primjer slike s kosim tekstom.

4.2.2. Ulazne datoteke

Slike opisane u 4.2.1 **ne predstavljaju** ulaz u sustav za određivanje strukture teksta. Nakon označavanja slika podatci o označavanju svake slike se izvoze i pohranjuju u datoteke u JSON formatu. Datoteke u JSON formatu predstavljaju nestrukturirani OCR-rezultat i ulaz u sustav. Ulazna datoteka u formatu JSON u kojoj su zapisani podaci o označavanju slike 4.3 prikazana je u isječku 4.3. Zbog specifičnosti sustava za označavanje slika i načina na koji pohranjuje informacije o označavanju, svi označeni znakovi bit će smješteni u jednu liniju u nasumičnom poretku i ta linija će biti smještena u jedan blok. Za svaki znak dostupna je informacija o Unicode vrijednosti znaka koja je smještena pod ključem `value`. Dodatno, za svaki znak dostupna je informacija o poziciji i veličini njegovog omeđujućeg pravokutnika (engl. *bounding box*). Za svaki

omeđujući pravokutnik poznate su sljedeće informacije:

- x - horizontalna pozicija gornjeg lijevog kuta,
- y - vertikalna pozicija gornjeg lijevog kuta,
- $width$ - širina i
- $height$ - visina.

Kako vrijednost x omeđujućeg pravokutnika raste tako je znak bliže desnom rubu slike. Kako vrijednost y omeđujućeg pravokutnika raste tako je znak bliže donjem rubu slike. Sve informacije o omeđujućem pravokutniku su vrijednosti iz skupa nene-gativnih realnih brojeva.

Sustav za označavanje slika izvozi još neke dodatne informacije o znakovima i o označenoj slici. Sve informacije koje nisu ovdje navede mogu se zanemariti.

Isječak 4.3: JSON datoteka s podatcima o označavanju slike 4.3.

```
1  {
2      "ocr_result": {
3          "blocks": [
4              {
5                  "lines": [
6                      {
7                          "chars": [
8                              {
9                                  "value": 83,
10                                 "bounding_box": {
11                                     "x": 4.548218,
12                                     "y": 271.68826,
13                                     "width": 12.136101,
14                                     "height": 22.48648
15                                 }
16                             },
17                             {
18                                 "value": 65,
19                                 "bounding_box": {
20                                     "x": 4.244385,
21                                     "y": 247.67685,
22                                     "width": 12.59581,
23                                     "height": 21.750519
24                                 }
25                             },
26                         // ostalih 388 znakova
27                     ]
28                 }
29             ]
```

```
30      }
31    ]
32  }
33 }
```

Izlaz iz sustava za određivanje strukture teksta

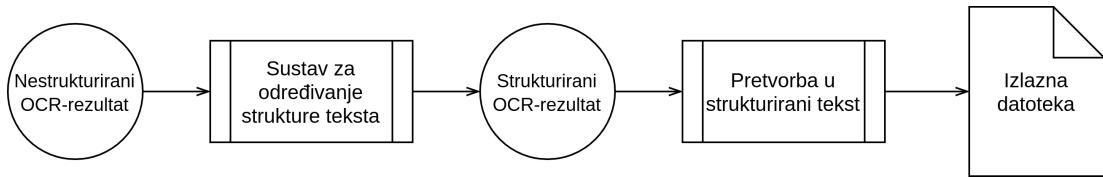
U odjeljku 4.1 navedeni su zahtjevi sustava i opisan je pojednostavljeni format izlaza iz sustava. Od sustava se očekuje da rezultat nakon određivanja strukture teksta prikaže u istom formatu u kojem je prikazan ulaz u sustav opisan u ovom odjeljku. Sve grupirane linije potrebno je smjestiti u jedan blok. Sustav ne treba isporučiti informacije koje su se mogu zanemariti na ulazu.

4.2.3. Očekivane izlazne datoteke

Očekivane izlazne datoteke predstavljaju ispravni strukturirani tekstualni sadržaj sa slike. Isječak 4.4 predstavlja ispravni strukturirani tekstualni sadržaj računa sa slike 4.3. Uočimo da u tekstualnom sadržaju ne postoje bjeline prije početka linije koje postoje u sadržaju na slici. Osim toga, višestruke bjeline u sadržaju na slici predstavljene su točno jednom bjelinom u tekstualnom sadržaju u datoteci.

Isječak 4.4: Tekstualni sadržaj slike 4.3.

```
1 POINTS TO $35 REWARD 8770
2 BALANCE REWARDS ACCT # ****2463
3 OPENING BALANCE 20820
4 EVERYDAY POINTS - RETAIL 410
5 CLOSING BALANCE 21230
6 ****
7 Walgreens 01875
8 ACCT 7681
9 SEQUENCE 1875220350
10 PAYMENT FROM PRIMARY
11 Get the flu shot that helps provide
12 a lifesaving vaccine to a child in need
13 Get a Shot. Give a Shot.® It's that easy
14 Learn more at the pharmacy.
```



Slika 4.6: Postupak dobivanja izlazne datoteke iz strukturiranog OCR-rezultata.

4.3. Korištenje skupa podataka za testiranje

Slika 4.6 prikazuje postupak dobivanja izlazne datoteke iz strukturiranog OCR-rezultata kojeg je vratio sustav za određivanje strukture teksta. Ulaz u sustav je ulazna datoteka u formatu JSON koja predstavlja nestrukturirani OCR-rezultat kao što je opisano u pododjeljku 4.2.2. Izlaz iz sustava je strukturirani OCR-rezultat u formatu JSON u kojemu se nalaze znakovi grupirani u linije i gdje su između riječi ubačeni znakovi bjeline. Strukturirani OCR-rezultat u formatu JSON potrebno je pretvoriti u izlaznu datoteku koja predstavlja strukturirani tekstualni sadržaj u formatu opisanom u pododjeljku 4.2.3 koji se onda uspoređuje sa očekivanim izlaznim datotekama dostupnim u skupu podataka za testiranje. Isječak 4.5 prikazuje pseudokôd algoritma za ispis OCR-rezultata na standradni izlaz.

Isječak 4.5: Pseudokôd algoritma za ispis OCR-rezultata na standardni izlaz.

```

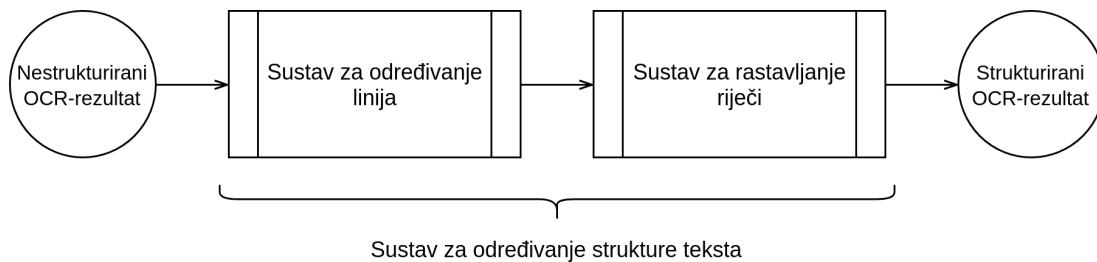
1 def ispisi(ocrRezultat)
2     for linija u ocrRezultat.linije
3         for znak u linija
4             print(znak.value)
5         end
6         print("\n")
7     end
8 end

```

Metode usporedbe izlaznih datoteka dobivenih iz strukturiranih OCR-rezultata i očekivanih izlaznih datoteka iz skupa podataka za testiranje biti će objašnjene u odjeljku.

5. Algoritmi za određivanje strukture teksta

Sustav za određivanje strukture teksta podijeljen je u dva podsustava: **sustav za određivanje linija** i **sustav za rastavljanje riječi**. Slika 4.6 prikazuje navedene komponente sustava i njihovu suradnju. Ulaz u sustav za određivanje linija je nestrukturirani OCR-rezultat dobiven od OCR-sustava koji je ujedno i ulaz u sustav za određivanje strukture teksta. Izlaz iz sustava za određivanje linija je OCR-rezultat u kojem su znakovi grupirani u linije kojima pripadaju. Izlaz iz sustava za određivanje linija je ulaz u sustav za rastavljanje riječi koji u svakoj liniji ubacuje na odgovarajuća mesta znakove bjeline koji odvajaju riječi. Izlaz iz sustava za rastavljanje riječi je ujedno i izlaz iz sustava za određivanje strukture teksta.

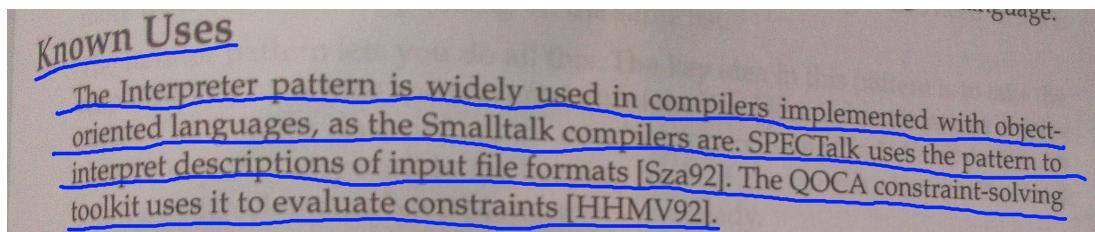


Slika 5.1: Komponente sustava za određivanje strukture teksta.

U nastavku ovog poglavlja objasniti ćemo implementirane algoritme za određivanje linija i za rastavljanje riječi, a u poglavlju bit će opisani rezultati i kratka analiza algoritama.

5.1. Algoritmi za određivanje linija

Algoritmi za određivanje linija trebaju na temelju omeđujućih pravokutnika svakog znaka znakove grupirati u linije. Slika 5.2 predstavlja vizualizaciju očekivanog rezultata algoritma za određivanje linija.



Slika 5.2: Vizualizacija detektiranih linija u sadržaju iz knjige.

Algoritam na ulazu prima nestrukturirani OCR-rezultat koji je ujedno i ulaz u sustav za određivanje strukture teksta. Izlaz iz algoritma je OCR-rezultat u kojem su znakovi grupirani u linije, sortirani s lijeva na desno i u kojem su linije sortirane tako da se najviša linija na slici nalazi na prvom mjestu.

5.1.1. Algoritam temeljen na maksimalnom preklapanju znakova

Algoritam za određivanje linija na temelju maksimalnog preklapanja znakova (u dalnjem tekstu: *algoritam*) temelji se na prepostavci da dva susjedna znaka koja se nalaze u istoj liniji ostvaruju maksimalno vertikalno preklapanje. Vertikalno preklapanje *overlap* dva znaka A i B definiramo izrazom:

$$\text{overlap}(A, B) := \frac{\max(0, \min(A_y + A_h, B_y + B_h) - \max(A_y, B_y))}{\min(A_h, B_h)} \quad (5.1)$$

Na početku svog rada algoritam uzlazno sortira sve znakove po horizontalnoj x vrijednost, zatim iterira po svakom znaku i konstruira linije gledajući s kojom linijom promatrani znak ostvaruje maksimalno preklapanje. Preklapanje s linijom definira se kao preklapanje sa zadnjim znakom u toj liniji. Promatrani znak pripadne onoj liniji s kojom ostvari maksimalno preklapanje:

$$l_{max} = \arg \max_{l \in L} \{ \text{overlap}(A, l_{-1}) \} \quad (5.2)$$

Skup svih dosad grupiranih linija označen je sa L , a l_{-1} označava zadnji znak u promatranoj liniji. Ako je znak s nekom linijom ostvario preklapanje vrijednosti 0 tada on postaje početak nove linije i tako se u skup L dodaje nova linija.

Rješavanje problema valovitih linija

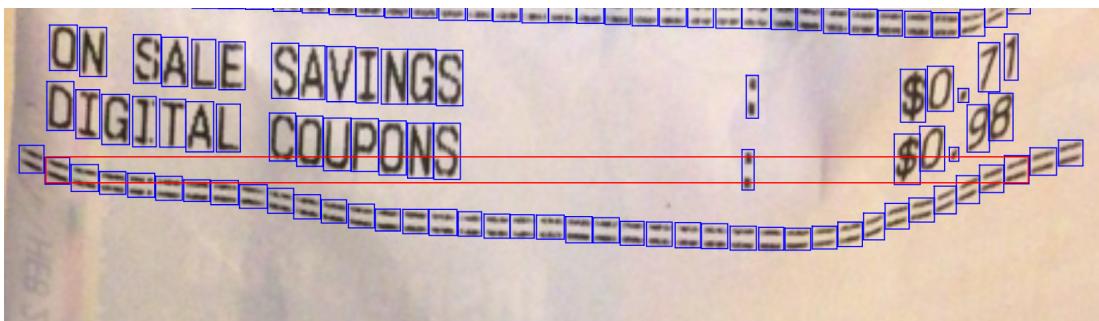
Budući da linije u sadržaju skupa podataka kojeg promatramo mogu biti valovite (slika 5.3), ponekad je poželjno izmjeriti preklapanje ne samo sa zadnjim znakom u liniji

nego i sa zadnjih nekoliko:

$$l_{max} = \arg \max_{l \in L, i \in [1, \min(|l|, c_1)]} \{overlap(A, l_{-i})\} \quad (5.3)$$

Za razliku od izraza 5.2 koji uzima u obzir samo zadnji znak u svakoj liniji, izraz 5.3 uzet će u obzir zadnjih $\min(|l|, c_1)$ znakova u svakoj liniji, gdje $|l|$ predstavlja duljinu linije, a c_1 predstavlja **parametar algoritma**. Eksperimentalno je utvrđeno da vrijednost parametra $c_1 = 1$ ostvaruje najbolje rezultate na sadržaju s računima iz trgovine, a da vrijednost parametra $c_1 = 2$ ostvaruje najbolje rezultate na sadržaju iz knjiga.

Slika 5.3 prikazuje kako znak = na desnoj strani crvenog pravokutnika ostvaruje maksimalno preklapanje sa znakom = na lijevoj strani crvenog pravokutnika koji nije zadnji znak u liniji u tom trenutku.



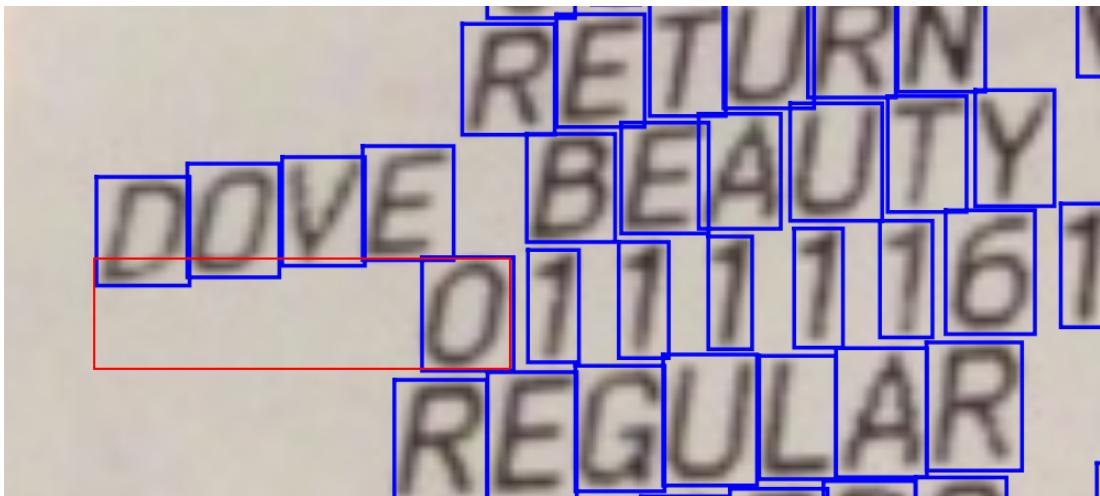
Slika 5.3: Valovite linije otežavaju određivanje linija.

Rješavanje problema preklapanja početaka dviju linija

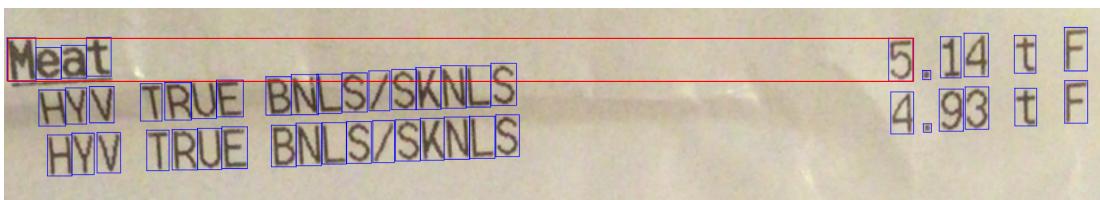
Slika 5.4 prikazuje kako je moguće preklapanje početaka dviju linija. Tako znakovi 0 i D ostvaruju nezanemarivo preklapanje čija će posljedica biti krivo određene linije. Kako bismo uspješno detektirali ovakve slučajevе uvodimo dodatan uvjet koji će odlučiti hoće li promatrani znak pripasti liniji s kojom ostvaruje maksimalno preklapanje:

$$\max_{i \in [1, \min(|l_{max}|, c_1)]} \{overlap(A, l_{max-i})\} > c_2 \quad (5.4)$$

Nakon što smo izrazom 5.3 odredili s kojom linijom promatrani znak ostvaruje maksimalno preklapanje trebamo provjeriti zadovoljava li iznos tog preklapanja uvijet naveden izrazom 5.4. Uvodimo novi parametar algoritma c_2 koji predstavlja donju granicu preklapanja koju znak mora ostvariti s linijom da bi joj se pripojio. Eksperimentalno je utvrđeno da vrijednost parametra $c_2 = 0,13$ ostvaruje najbolje rezultate na sadržaju s računima iz trgovine, a da vrijednost parametra $c_2 = 0,05$ ostvaruje najbolje rezultate na sadržaju iz knjiga.



Slika 5.4: Preklapanje početaka dviju linija.



Slika 5.5: Lažno pozitivno preklapanje

Rješavanje problema lažno pozitivnog preklapanja

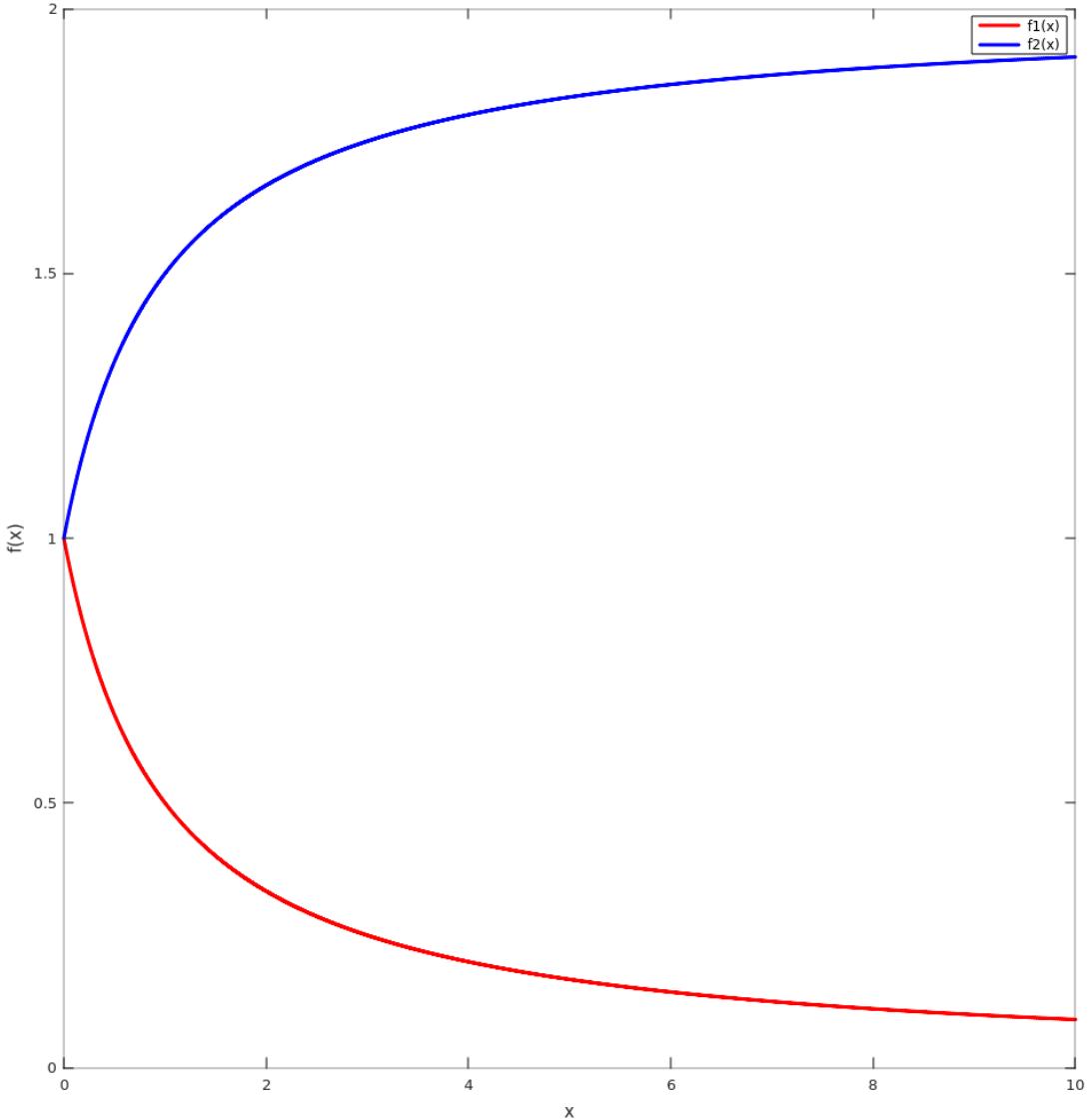
Slika 5.5 prikazuje slučaj kada znak pripadne krivoj liniji zato jer s njom ostvaruje maksimalno preklapanje zbog zakriviljenosti slike i sadržaja. U ovom primjeru znak 5 pripasti će prvoj liniji `Meat`, a trebao bi pripasti drugoj liniji. Promatrajući skup podataka za koji rješavamo problem zaključili smo kako uvijek treba dati prednost onoj liniji čiji je zadnji znak bliži znaku kojeg promatramo. Tako će u primjeru sa slike 5.5 znak 5 pripasti drugoj liniji zato jer je njezin zadnji znak bliži znaku 5 nego znak `t` iz prve linije i zato jer s zadnjim znakom u drugoj liniji ipak postiže nezanemarivo preklapanje. Da bismo uspješno rješili navedeni problem ćemo dvije funkcije i dva dodatna parametra algoritma. Definirajmo prvo normaliziranu udaljenost između dva znaka:

$$d(A, B) = \frac{|A_x - B_x|}{\min(A_w, B_w)} \quad (5.5)$$

Zatim ćemo definirati navedene dvije funkcije (slika 5.6) koje koriste dva nova parametra algoritma c_3 i c_4 :

$$f_1(x) = \frac{1}{1 + c_3 \cdot x} \quad (5.6)$$

$$f_2(x) = 1 + \frac{c_4 \cdot x}{1 + c_4 \cdot x} \quad (5.7)$$



Slika 5.6: Grafovi funkcije f_1 (crveno) i funkcije f_2 (plavo) za parametre $c_3 = c_4 = 1$.

Ideja je da koristimo navedene funkcije f_1 i f_2 kako bismo linijama olakšali odnosno otežali postizanje preklapanja u ovisnosti o udaljenosti između njihovih zadnjih znakova. Recimo da u smo u postupku traženja maksimalnog preklapanja u jednom trenutku ostvarili maksimalno preklapanje s linijom l_1 iznosa p_1 . S idućom linijom l_2

izmjerimo preklapanje p_2 . Da bi linija l_2 postigla novo maksimalno preklapanje sa promatranim znakom mora vrijediti $p_2 > p_1$. Međutim, recimo da smo uočili da je zadnji znak linije l_2 bliži promatranom znaku nego zadnji znak linije l_1 . Koristeći navedenu opservaciju možemo liniji l_2 olakšati postizanje novog maksimalnog preklapanja tako da ostvarno preklapanje linije l_1 umanjimo za neki iznos. Budući da je zadnji znak linije l_2 bliži promatranom znaku novi uvijet koji linija l_2 mora zadovoljiti da bi ju smatrali novim maksimalnim preklapanjem glasi:

$$p_2 > p_1 \cdot f_1(d(l_{1-1}, l_{2-1})) \quad (5.8)$$

Funkcija f_1 će liniji l_2 pomoći tim više što su zadnji znakovi linija l_1 i l_2 udaljeniji. Da je zadnji znak linije l_2 bio dalje od promatranog znaka nego zadnji znak linije l_1 onda bismo liniji l_2 trebali otežati postizanje novog maksimalnog preklapanja čak i ako vrijedi $p_2 > p_1$. U tom slučaju novi uvijet za liniju l_2 glasi:

$$p_2 > p_1 \cdot f_2(d(l_{1-1}, l_{2-1})) \quad (5.9)$$

Funkcija f_2 će liniji l_2 otežati tim više što su zadnji znakovi linija l_1 i l_2 udaljeniji. Eksperimentalno je utvrđeno da vrijednosti parametra $c_3 = c_4 = 0,13$ ostvaruje najbolje rezultate na sadržaju s računima iz trgovine i na sadržaju iz knjiga.

Nakon što smo svaki znak smjestili u odgovarajuću liniju, linije je potrebno sortirati uzlazno po y vrijednosti prvih znakova linije. Naime, u početnom koraku sortiranja znakova po x vrijednosti moguće je na primjer da prvi znak u sortiranim znakovima započinje neku liniju u sadržaju koja nije prva. Budući da će taj znak u našem algoritmu biti detektiran kao početak nove linije to će liniju koju taj znak započinje smjestiti na početak, a ne na mjesto kojem ta linija pripada.

Konačno, pseudokôd algoritma za određivanje linija na temelju maksimalnog preklapanja znakova prikazan je u isječku. U navedenom isječku funkcije f_1 i f_2 implicitno koriste vrijednosti parametara $c3 = c4 = 0,13$.

Isječak 5.1: Pseudokôd algoritma za određivanje linija.

```

1 def odrediLinije(nestrukturiraniOcrRezultat)
2     znakovi = sortirajPoX(nestrukturiraniOcrRezultat.sviZnakovi)
3     linije = []
4
5     for znak u znakovi
6         maxPreklapanje = 0
7         linijaSaMaxPreklapanjem = null
8         for linija u linije
9             p = 0

```

```

10      tezina = 1
11      for i u [0, min(c1, linija.duljina)]
12          p = max(p, overlap(znak, linija[-i]))
13      end
14      if linijaSaMaxPreklapanjem != null
15          if linijaSaMaxPreklapanjem[-1].x < linija[-1].x
16              tezina = f1(d(linijaSaMaxPreklapanjem[-1], linija[-1]))
17          else
18              tezina = f2(d(linijaSaMaxPreklapanjem[-1], linija[-1]))
19          end
20      end
21      if p > maxPreklapanje * tezina
22          maxPreklapanje = p
23          linijaSaMaxPreklapanjem = linija
24      end
25  end
26
27  if maxPreklapanje > c2
28      linijaSaMaxPreklapanjem.nadodaj( znak )
29  else
30      linije.nadodaj([znak])
31  end
32 end
33
34 sortiraneLinije = sortirajPoY(linije)
35
36 return new OcrRezultat(sortiraneLinije)
37 end

```

5.2. Algoritmi za rastavljanje riječi

Algoritmi za rastavljanje riječi na ulaz primaju OCR-rezultat u kojemu su znakovi grupirani u linije u prethodnom koraku opisanom u odjeljku 5.1. Zadaća algoritama za rastavljanje riječi je da između znakova, za koje smatra da su završetak prethodne odnosno početak iduće riječi, ubaci znak bjeline čija će vrijednost (engl. *value*) biti 32, a ostale informacije mogu biti proizvoljne.

U nastavku ovog odjeljka opisati ćemo tri algoritma za rastavljanje riječi:

- algoritam temeljen na prosječnoj širini znaka
- algoritam temeljen na prosječnoj relativnoj udaljenosti
- algoritam temeljen na prosječnoj udaljenosti centara

5.2.1. Algoritam temeljen na prosječnoj širini znaka

Algoritam temeljen na prosječnoj širini znaka najjednostaviji je od svih algoritama koje ćemo u ovom odjeljku opisati. Algoritam se temelji na pretpostavci da je širina razmaka između riječi proporcionalna sa prosječnom širinom znakova u promatranoj liniji. Označimo prosječnu širinu znaka u liniji l_i sa \overline{w}_i :

$$\overline{w}_i = \frac{\sum_{A \in l_i} A_w}{|l_i|} \quad (5.10)$$

Sada možemo postaviti uvijet koji će odlučiti treba li ubaciti bjelinu između dva znaka A i B :

$$|A_x - (B_x + B_w)| > \overline{w}_i \cdot c_1 \quad (5.11)$$

Parametar c_1 je jedini **parametar algoritma** za koji smo eksperimentalno utvrdili da najbolje rezultate za sadržaj na računima iz trgovine ostvaruje vrijednost 0,8, a najbolje rezultate za sadržaj iz knjiga ostvaruje vrijednost 0,44. Isječak 5.2 prikazuje pseudokôd algoritma za rastavljanje riječi temeljenog na prosječnoj širini znaka. Izraz s lijeve strane nejednadžbe 5.11 označava udaljenost između lijevog ruba znaka A i desnog ruba znaka B s prepostavkom da je znak A desno od znaka B , odnosno da vrijedi $A_x > B_x$.

Isječak 5.2: Pseudokôd algoritma za rastavljanje riječi temeljen na prosječnoj širini znaka.

```

1  def rastaviRijeci(ocrRezultat)
2      for linija u ocrRezultat.linije
3          prosjecnaSirina = 0
4          for znak u linija
5              prosjecnaSirina += znak.width
6          end
7          prosjecnaSirina /= linija.duljina
8
9          i = 0
10         j = 1
11         while j < linija.duljina
12             d = linija[j].x - (linija[i].x + linija[i].width)
13             if d > prosjecnaSirina * c1
14                 bjelina = new Znak()
15                 bjelina.value = 32
16                 bjelina.x = linija[i].x + linija[i].width
17                 bjelina.y = linija[i].y
18                 bjelina.width = d
19                 bjelina.height = linija[i].height

```

```

20     linija.ubaciZnakPrijePozicije(j, znak)
21     i += 2
22     j += 2
23 else
24     i++
25     j++
26 end
27 end
28
29
30 return ocrRezultat
31 end

```

Algoritam prikazan pseudokôdom 5.2 ubacuje znak bjeline čija je vrijednost jednaka 32, a ostale informacije su određene na sljedeći način:

- x - horizontalna pozicija gornjeg desnog kuta lijevog znaka,
- y - vertikalna pozicija gornjeg lijevog kuta desnog znaka,
- $width$ - horizontalna udaljenost između gornjeg desnog kuta lijevog znaka i gornjeg lijevog kuta desnog znaka,
- $height$ - visina lijevog znaka.

Pod *lijevi znak* misli se na znak u liniji na poziciji i , a pod *desni znak* misli se na znak u liniji na poziciji j . U liniji 20 ubacujemo novi znak bjeline prije znaka na poziciji j što ima za posljedicu pomicanje svih elemenata, desno od j , uključujući i element na poziciji j , za jedno mjesto udesno. U liniji 12 algoritam prepostavlja da je znak na poziciji j desno od znaka na poziciji i .

6. Zaključak

LITERATURA

Matthew Christy, Anshul Gupta, Elizabeth Grumbach, Laura Mandell, Richard Furuta, i Ricardo Gutierrez-Osuna. Mass digitization of early modern texts with optical character recognition. *J. Comput. Cult. Herit.*, 11(1):6:1–6:25, Prosinac 2017. ISSN 1556-4673. doi: 10.1145/3075645. URL <http://doi.acm.org/10.1145/3075645>.

Filip Gulan. Očitavanje rukom pisanih slova. Završni rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Unska 3, 10000 Zagreb, Republika Hrvatska, Lipanj 2016.

Gaurav Gupta, Shobhit Niranjan, Ankit Shrivastava, i R Mahesh K Sinha. Document layout analysis and classification and its application in ocr. U *Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW'06. 10th IEEE International*, stranice 58–58. IEEE, 2006.

Abdeslam El Harraj i Naoufal Raissouni. OCR accuracy improvement on document images through a novel pre-processing approach. *CoRR*, abs/1509.03456, 2015. URL <http://arxiv.org/abs/1509.03456>.

Noman Islam, Zeeshan Islam, i Nazia Noor. A survey on optical character recognition system. *CoRR*, abs/1710.05703, 2017. URL <http://arxiv.org/abs/1710.05703>.

Ivan Jurin. Višeobjektni modeli detekcije za raspoznavanje teksta dubokim učenjem. Diplomski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Unska 3, 10000 Zagreb, Republika Hrvatska, Lipanj 2017.

Sukhpreet Kaur i Simpel Rani. A survey on feature extraction and classification techniques for character recognition of indian scripts. 2016.

Alex Krizhevsky, Ilya Sutskever, i Geoffrey E Hinton. Imagenet classification with

deep convolutional neural networks. In *Advances in neural information processing systems*, stranice 1097–1105, 2012.

Rayson Laroca, Evair Severo, Luiz A. Zanlorensi, Luiz S. Oliveira, Gabriel R. Gonçalves, William R. Schwartz, i David Menotti. A robust real-time automatic license plate recognition based on the YOLO detector. *CoRR*, abs/1802.09567, 2018. URL <http://arxiv.org/abs/1802.09567>.

Gurpreet S Lehal i Chandan Singh. Feature extraction and classification for ocr of gurmukhi script. *VIVEK-BOMBAY-*, 12(2):2–12, 1999.

Jisheng Liang, Jaekyu Ha, Robert M Haralick, i Ihsin T Phillips. Document layout structure extraction using bounding boxes of different entitles. In *Applications of Computer Vision, 1996. WACV'96., Proceedings 3rd IEEE Workshop on*, stranice 278–283. IEEE, 1996.

Yi-Feng Pan, Xinwen Hou, i Cheng-Lin Liu. A hybrid approach to detect and localize texts in natural scene images. *IEEE Transactions on Image Processing*, 20(3):800–813, 2011.

Hamed Saghaei. Proposal for automatic license and number plate recognition system for vehicle identification. *CoRR*, abs/1610.03341, 2016. URL <http://arxiv.org/abs/1610.03341>.

Sarah Schulz i Jonas Kuhn. Multi-modular domain-tailored ocr post-correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, stranice 2716–2726, 2017.

Shangxuan Tian, Yifeng Pan, Chang Huang, Shijian Lu, Kai Yu, i Chew Lim Tan. Text flow: A unified text detection system in natural scene images. *CoRR*, abs/1604.06877, 2016. URL <http://arxiv.org/abs/1604.06877>.

Abhishek Verma, Suket Arora, i Preeti Verma. Ocr-optical character recognition. In *7th International Conference on Recent Innovations in Science, Engineering and Management*, 2016.

Rohit Verma i Jahid Ali. A-survey of feature extraction and classification techniques in ocr systems. *International Journal of Computer Applications & Information Technology*, 1(3):1–3, 2012.

Ivo Vynckier. How ocr works, a close look at optical character recognition, 2018. URL <http://how-ocr-works.com/OCR/OCR.html>. Přistupano: 01.06.2018.

Christoph Wick, Christian Reul, i Frank Puppe. Improving OCR accuracy on early printed books using deep convolutional networks. *CoRR*, abs/1802.10033, 2018. URL <http://arxiv.org/abs/1802.10033>.

Fei Yin i Cheng-Lin Liu. Handwritten text line extraction based on minimum spanning tree clustering. U *Wavelet Analysis and Pattern Recognition, 2007. ICWAPR'07. International Conference on*, svezak 3, stranice 1123–1128. IEEE, 2007.

Xu-Cheng Yin, Xuwang Yin, i Kaizhu Huang. Robust text detection in natural scene images. *CoRR*, abs/1301.2628, 2013. URL <http://arxiv.org/abs/1301.2628>.

Weiheng Zhu, Yuanfeng Liu, i Liang Hao. A novel ocr approach based on document layout analysis and text block classification. U *Computational Intelligence and Security (CIS), 2016 12th International Conference on*, stranice 91–94. IEEE, 2016.

Sustav za određivanje strukture teksta na temelju položaja pojedinih znakova

Sažetak

Ključne riječi:

Text Layout Analysis System Based on Individual Character Positions

Abstract

Keywords: