Detekcija napada koji mijenjaju izgled web-sjedišta analizom slika sjedišta

Milin, Noa

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1472

DETEKCIJA LAŽNIH SJEDIŠTA WEBA ANALIZOM SLIKA NASLOVNICA

Noa Milin

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1472

DETEKCIJA LAŽNIH SJEDIŠTA WEBA ANALIZOM SLIKA NASLOVNICA

Noa Milin

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1472

Pristupnik:	**Noa Milin (0036527811)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	izv. prof. dr. sc. Marin Vuković

Zadatak: **Detekcija lažnih sjedišta weba analizom slika naslovnica**

Opis zadatka:

Tražilice i katalozi imaju cilj analize i kategorizacije sjedišta weba. Međutim, upravljanje katalozima je dužnost administratora te postoji sklonost ljudskoj pogrešci. Dodatan problem jest promjena sadržaja ili vlasništva sjedišta, kao kompromitacija. Jedno od rješenja koje bi pomoglo upravljanju katalogom jest analiza slika naslovnica sjedišta weba. Analizirajte rješenja za usporedbu slika, s naglaskom na slike naslovnica sjedišta weba. Na temelju analize predložite i implementirajte model za detekciju anomalija u slikama kako bi se moglo ustanoviti koja sjedišta su kompromitirana odnosno koja sjedišta ne pružaju sadržaj koji odgovara deklariranom naslovu sjedišta. Kao izvor slika naslovnica koristite početnu stranicu Republike Hrvatske WWW.HR.

Rok za predaju rada: 14. lipnja 2024.

SADRŽAJ

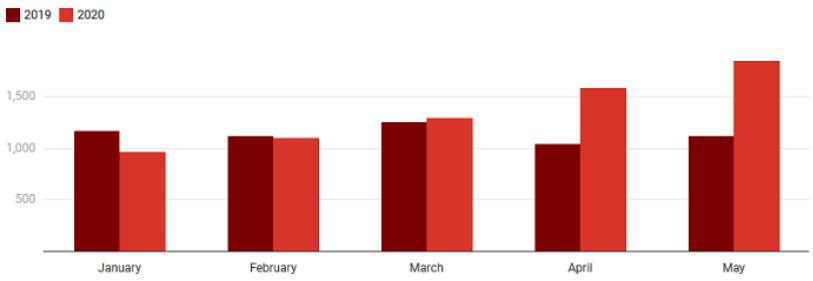
1. Uvod

Tražilice i katalozi predstavljaju velik izvor kategoriziranih sjedišta weba. Međutim, broj web sjedišta u katalozima nerijetko nadmašuje onaj koji je isplativo ručno nadzirati, što vrijedi i za katalog početne stranice Republike Hrvatske WWW.HR. Naime, katalog ne sadrži aktualne informacije o web sjedištima, već one koje su aktualne bile na datum dodavanja u katalog. Neki od mogućih problema kojima su podložna tako pohranjena web sjedišta su promjena sadržaja ili vlasništva te kompromitacija. Web sjedišta pohranjena u katalogu kroz to vrijeme mogu biti podložna promjeni sadržaja ili vlasništva. Promjenom vlasništva sadržaj se često mijenja zbog čega opis iz kataloga može postati zastario. Čak i uz isto vlasništvo, sadržaj može biti promijenjen do te mjere da opis iz kataloga više nije ažuran. Takvi slučajevi ukazuju na lažno web sjedište. Jedan od primjera kompromitacije web sjedišta je defacement napad. Kod takvog napada mijenja se ili briše originalni sadržaj web sjedišta, što često uključuje promjenu naslovne stranice web sjedišta. Uzevši u obzir navedene izazove u održavanju kataloga, samo upravljanje katalogom od strane administratora nije dovoljno te je bitno uspostaviti druge načine provjere valjanosti pohranjenih web sjedišta.

Cilj rada je korištenjem slike naslovnice i pohranjenog opisa sjedišta weba prepoznati je li sjedište weba lažno. Rješenje je predstavljeno sakupljanjem slika i informacija o sjedištu weba web paukom te naknadnim treniranjem feed-forward neuronske mreže na tim podacima. Tako dobiven model služio bi za detekciju lažnih web sjedišta pohranjenih u katalogu početne stranice Republike Hrvatske WWW.HR.

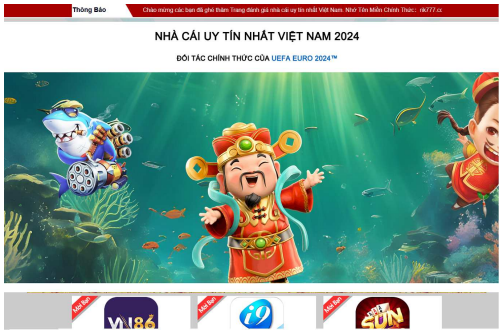
2. Lažna sjedišta weba

2.1. Defacement

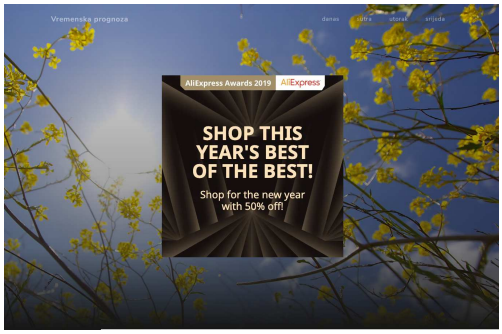


Slika 2.1: Porast defacement napada tijekom pandemije [6]

Defacement je tip napada koji mijenja ili briše originalni vizualni sadržaj stranice. Generalna populacija upoznata je s ovakvim napadima djelovanjem poznatih haktivističkih grupa kao što su *Anonymous* ili *Lizard Squad*. Međutim, glavni razlog naglog porasta defacement napada nisu poznate grupe, već pojava novih i neiskusnih hakera, tzv. *Script Kiddies*[14]. Posljednjih godina, a pogotovo tijekom brojnih lockdown-ova povodom COVID-19 pandemije, znatno se povećao broj defacement napada na web-sjedišta (Slika 2.1).



(a) [24]

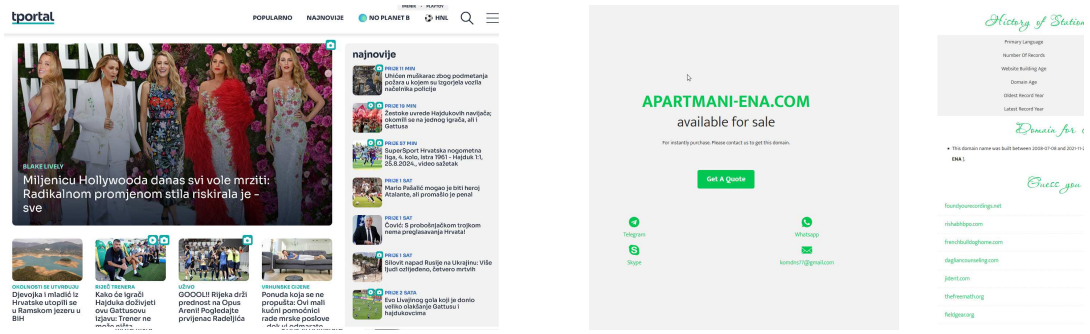


(b) [23]

Slika 2.2: Web sjedišta pohranjena na WWW.HR katalogu koja su žrtva defacement napada

Defacement napadi u većini slučajeva ne utječu na sam URL stranice jer je međuostalom cilj napadača obmanuti korisnika stranice lažnim sadržajem na već poznatoj stranici. Sadržaj web sjedišta koje je žrtva defacementa može biti sasvim drugačiji od originalnog sadržaja (Slika 2.2a), ali postoje i slučajevi kad napadač ostavi dio originalnog sadržaja kako bi se bolje osigurao od detekcije (Slika2.2b).

2.2. Promjena u sadržaju i vlasništvu



(a) Promjena sadržaja [31] **(b)** Promjena sadržaja i vlasništva [22]

Slika 2.3: Promjena sadržaja i vlasništva

Promjene u sadržaju web sjedišta ne moraju nužno značiti da je stranica kompromitirana, ali to sjedište i dalje može biti lažno ili zastarjelo. Npr. jedno web sjedište (Slika2.3a) u katalogu reklamirano na poddomeni koja više ne postoji. U takvom slučaju izvorna stranica više nije dostupna, a korisnik će biti preusmjeren na glavnu domenu koja nije kompromitirana. Međutim reklamirani sadržaj nije prisutan pa stoga to web sjedište deklariramo lažnim. Takva je sjedišta teško raspoznati samom analizom slike naslovnice te je sam kontekst opisa sjedišta u katalogu bitniji.

Drugi primjer je slučaj kada je URL u katalogu isti kao i stvarni URL, ali se sadržaj od vremena dodavanja u katalog promijenio (Slika2.3b). Sadržaj je također izmijenjen te umjesto reklamiranog sadržaja ističe kako je domena na prodaju. Ovakvi primjeri često ukazuju na promjenu vlasništva, pri čemu je originalni vlasnik napustio web sjedište ili ga prodao. Prijašnje i trenutno vlasništvo za dosta se stranica može provjeriti na stranici *who.is*[21]. Sadržaj web sjedišta, odnosno starije slike naslovne stranice možemo u nekim slučajevima dobiti pomoću stranice *Wayback Machine*[20]

Također postoje situacije kada web sjedište postane neaktivno i na zahtjev korisnika vraća određene HTTP status kodove. Neka od takvih sjedišta će nam i dati sadržaj,

ali on će sigurno odstupati od onoga u katalogu. Takvo ponašanje može implicirati različite tehničke probleme ili promjene na stranici. Od napuštanja web sjedišta od strane vlasnika do predaje novom vlasniku koji je domenu ostavio zapuštenom. Dobar alat za efikasno provjeravanje status kodova je *Postman*[13]. U njemu možemo slati raznorazne zahtjeve stranicama i međuostalom dobiti status kod.

Neki od status tih kodova uočeni na sjedištima kataloga WWW.HR:

- **400 (Bad Request)** Poslužitelj nije mogao odgovoriti na zahtjev zbog pogrešne sintakse ili podataka.
- **401 (Unauthorized)** Za pristup je potrebna autorizacija, što može značiti da je stranica zaštićena lozinkom ili da je pristup ograničen.
- **403 (Forbidden)** Pristup stranici je zabranjen, što ukazuje na ograničenja od strane vlasnika ili poslužitelja.
- **404 (Not Found)** Stranica više ne postoji na poslužitelju, što ukazuje na to da je sadržaj uklonjen ili premješten.
- **410 (Gone)** Stranica je trajno uklonjena.
- **500 (Internal Server Error)** Poslužitelj je naišao na nepredviđeni problem, što ukazuje na tehničke poteškoće.
- **502 (Bad Gateway)** Poslužitelj je primio nevažeći odgovor od nadređenog poslužitelja, dakle došlo je do problema u komunikaciji.
- **503 (Service Unavailable)** Poslužitelj trenutno nedostupan zbog održavanja ili preopterećenja.
- **504 (Gateway Timeout)** Poslužitelj nije u propisanom vremenu dobio odgovor od nadređenog poslužitelja, što ukazuje na spor odziv ili tehničke poteškoće oko mreže.
- **522 (Connection Timed Out)** Poslužitelj nije uspio odgovoriti na zahtjev unutar zadanog vremena, što ukazuje na preopterećenje ili tehničke poteškoće.
- **525 (SSL Handshake Failed)** Nije moguće uspostaviti sigurnu vezu s poslužiteljem.
- **530 (Site is Frozen)** Stranica je privremeno zamrznuta, često zbog neaktivnosti unutar nekog vremena.

3. Web pauk za analizu sjedišta weba

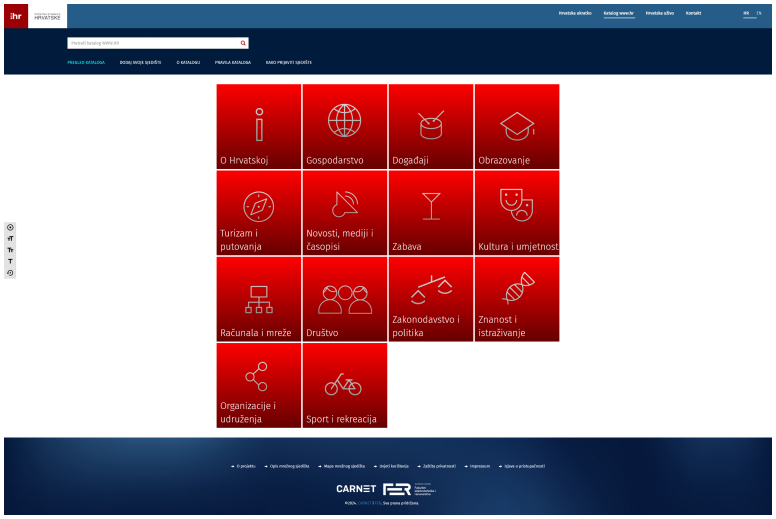
3.1. Općeniti web pauk

Web pauci (engl. *Web Crawlers*) programi su koji služe za automatsko pretraživanje web stranica. Njihova primarna svrha je dohvaćanje web stranica, praćenje linkova te prikupljanje podataka s posječenih stranica. Primjenjuju se u raznim zadacima kao što su npr. indeksiranje pretraživača i *data mining*[2]. Web pauci funkcioniraju tako da šalju HTTP zahtjeve poslužitelju, primaju odgovore i ovisno o implementaciji sakupljaju podatke iz HTML-a stranice.

3.2. Implementacija web pauka *kataloger.py*

Programski kod web pauka napisan je u programskom jeziku *Python* koristeći radni okvir *Scrapy*. *Scrapy* je radni okvir koji se generalno koristi za sakupljanje podataka s web stranica. Nudi različite implementacije web paukova, od kojih je najčešće korištena varijanta *CrawlSpider*[18]. On nam nudi jednostavno praćenje i prelaženje između linkova na stranicama. Linkovi koje pauk treba ili ne treba pratiti definiraju se pravilima unutar samog koda pauka. Iako je *CrawlSpider* jedan od jednostavnijih vrsta *Scrapy* paukova, proširenjem njegovih osnovnih funkcionalnosti možemo ostvariti naprednijeg web pauka, korisnijeg za nekakav konkretni zadatak.

Pauk implementiran u sklopu ovog rada postavljen je da obilazi isključivo katalog početne stranice Republike Hrvatske WWW.HR (Slika 3.1) na domeni *"www.hr"*. Razlog tome leži u specifičnom procesu dohvaćanja potrebnih podataka iz HTML-a stranica unutar tog kataloga. Naime, dohvat podataka je dio implementacije pauka koji može varirati ovisno o domeni koju pauk obilazi. Ovi podaci koristit će se za kreiranje skupa podataka potrebnih za nadzirano treniranje neuronske mreže. Detaljan opis jednog takvog pauka, implementiranog za potrebe ovog rada, bit će prikazan u narednim poglavljima.



Slika 3.1: Početna stranica kataloga WWW.HR s prikazanim osnovnim kategorijama [25]

3.2.1. Osnovne funkcionalnosti

Kako bi se pauku omogućio obilazak stranice, prvo je potrebno zadati domenu koju smije obilaziti i početni URL od kojeg će obilazak krenuti. Dozvoljena domena postavlja se u listi *allowed _domains*, dok se početni URL postavlja u listi *start_urls*. Također, pauku se dodjeljuje naziv putem varijable *name*.

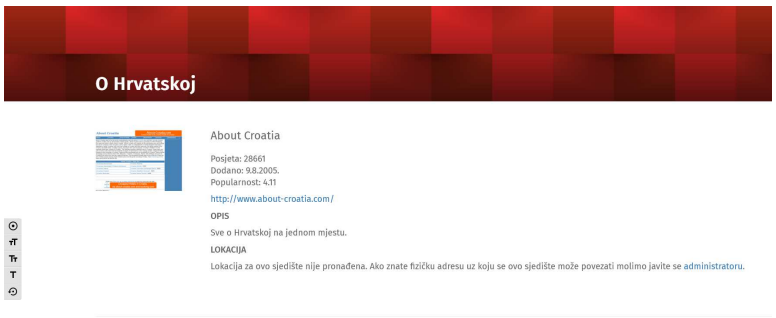
```
class KatalogerSpider(CrawlSpider):
    name = "kataloger"
    allowed_domains = ["www.hr"]
    start_urls = ["https://www.hr/katalog"]
    # Ostatak koda
```

Nakon što pauku zadamo odakle kreće i koje domene smije posjećivati, potrebno je definirati pravila koja određuju koje linkove pauk može pratiti, a koje ne. To se postiže atributom *Rules*.

Rules je lista unutar razreda *CrawlSpider* čiji su elementi pravila koja definiramo. Pravila se zadaju objektom tipa *LinkExtractor* koji određuje koje će linkove pauk pratiti sa svake stranice koju obiđe. Svako sjedište weba pohranjeno u katalogu WWW.HR (Slika 3.1) ima vlastitu stranicu koja je smještena u specifičnoj kategoriji, odnosno potkategoriji kataloga. Te stranice sadrže neke osnovne podatke o svakom sjedištu.

```
rules = (
    Rule(LinkExtractor(
        allow=r"[^/]+(/[^/]+)*/index\.hr\.html$")),
    Rule(LinkExtractor(
        allow=r"sjediste/"),
        callback="parse_item"),
    )
```

Cilj pauka je obići svaku takvu stranicu i sakupiti određene podatke. Kako bismo to postigli prvo je potrebno definirati pravilo koje pauku omogućava obilazak svih stranice kategorija i potkategorija kataloga. Putanja stranice kategorija i potkategorija ima oblik: *"/X/Y/Z/index.hr.html"* gdje "X" označava glavnu kategoriju, a "Y" i "Z" potkategorije (može ih biti više ili manje od dvije). To je izvedeno prvim pravilom unutar liste *Rules*. Dio *allow*= koristi se za određivanje linkova koje pauk smije pratiti, a u ovom slučaju koristi se regularni izraz koji pokriva sve moguće kategorije.



Slika 3.2: Sjedište s URL-om čija putanja završava sa *"sjediste/21555*" [27]

Nakon toga zadajemo pauku pravilo kojim se dopušta praćenje stranica sjedišta pohranjenih po tim kategorijama. Putanja tih stranica završava u formatu *"sjedište/X"*, gdje je "X" jedinstven broj za svako sjedište u katalogu. Drugo pravilo u listi *Rules* pauku dozvoljava obilazak tih stranica (Slika 3.2).

Također, potrebno je prikupiti podatke prilikom obilaska stranica pa se unutar drugog pravila dodaje *callback="parse_item"*. Callback funkcija se poziva za svaki link koji pauk prati prema tom pravilu, a time se poziva *parse_item* funkcija koja prima *response* objekt kao argument. Pomoću *response* objekta, unutar funkcije *parse_item* prikupljaju se traženi podaci sa stranice.

Unutar *Scrapy* projekta postoji i datoteka *settings.py*. Tamo je omogućena prilagodba paukovog ponašanja zahtjevima zadatka.

Postavke koje definiraju ponašanje pauka također možemo zadati unutar razreda *CrawlSpider* koristeći rječnik *custom_settings*. Možemo zadati nove postavke ili proširiti već postojeće koje se nalaze u *settings.py*. Dodane su dvije nove postavke.

```
custom_settings = {
    "DOWNLOAD_TIMEOUT": 10,
    "FEEDS": {
     # ostatak koda
```

Prva je *DOWNLOAD_TIMEOUT: 10* koja određuje da pauk ima najviše deset sekundi da preuzme stranicu, odnosno link do kojeg je došao slijedeći jedno od pravila opisanih ranije. Postavka je dodana kako bi se pauku omogućilo dulje, ali ne i predugo čekanje na odgovor web stranice.

Druga postavka je *"FEEDS" : {...}*. Unutar ovog rječnika definira se gdje i kako će pauk spremati podatke koje sakupi tijekom obilaska. Moguće je zadati format datoteke (npr. *CSV* ili *JSON*) u koju se spremaju podaci te sakupljeni podaci koji će biti spremljeni u tu datoteku. Detaljna izvedba struktura podataka bit će opisana kasnije.

Ranije spomenuta callback funkcija *parse_item* ključna je u izvedbi pauka. U njoj sakupljamo podatke koji će se kasnije koristiti u drugom callbacku *parse_actual_url*.

```
def parse_item(self, response):
    item_loader = ItemLoader(item=\emph{WwwItem}(),
                             response=response)
    item_loader.add_css( ... )
    item_loader.add_value( ... )
    # ...
    # Npr. jedna default vrijednost
    item_loader.add_value("html_match", "[FAILURE]")
    # ...
    item = item_loader.load_item()
    supposed_url = response.css(
        # neki CSS selektor
        ).get()
    yield response.follow(
        # argumenti
    )
```

Kako bismo unutar funkcije *parse_item* prvo sakupili podatke sa sjedišta, koristimo *ItemLoader*. On inicijalizira objekt tipa *WwwItem*. Da bismo to izveli, stvaramo *item_loader* koji prima objekt tipa *WwwItem* i *response* objekt, koji je argument funkcije *parse_item* te sadrži podatke dohvaćene sa stranice sjedišta.

```
class WwwItem(scrapy.Item):
    supposed_site_title = scrapy.Field(
        input_processor=MapCompose(remove_tags,
                                   clean_data),
        output_processor=TakeFirst(),
    )
    # Ostala polja
```

Razred *WwwItem* definiran je unutar datoteke *items.py* u *Scrapy* projektu. Ovaj razred sadrži polja koja su objekti tipa *Field*. Svako od tih polja predstavlja određeni podatak o sjedištu koji želimo da pauk sakupi iz HTML-a stranice ili putem izračuna nekih funkcija. Za svako polje definirane su funkcije za obradu ulaznih podataka (*input_processor*) i formatiranje izlaznih podataka (*output_processor*), koje osiguravaju da se podaci pravilno obrade i upišu u CSV datoteku.

Ta se polja popunjavaju unutar funkcije *parse_item* putem *item_loader-a*, koji koristi funkcije *add_css* i *add_value*. Obje funkcije kao argument primaju naziv polja koji će popuniti te vrijednost koja će biti pridodana tom polju. Funkcija *add_css* koristi CSS selektore kako bi pristupila HTML elementima stranice i dohvatila vrijednosti, dok *add_value* izravno dodaje unaprijed definirane vrijednosti.

Vrijednosti polja ne moraju uvijek biti uspješno prikupljene. Takvi su slučajevi pokriveni u *custom_settings* dijelu koda unutar rječnika *FEEDS*. Tamo je definirano da će polja bez uspješno prikupljenih podataka dobiti vrijednost *None*. Međutim, za neka polja definirane su i druge, unaprijed zadane vrijednosti, koje se postavljaju kako bi se omogućila daljnja obrada u funkciji *parse_actual_url*. Takva polja uključuju *layout_score*, *contrast_score*, i druge, a njihova će vrijednost biti ažurirana u narednim fazama parsiranja.

Nakon što su pridodane vrijednosti svih polja, kreiramo objekt tipa *WwwItem* putem *item_loader-a* koji poziva funkciju *load_item*. Zatim se putem CSS selektora dohvaća link koji vodi na stranicu opisanu u sjedištu.

Na kraju, koristimo *yield*, koja prati prethodno navedeni link putem funkcije *follow*. Argumenti *follow* funkcije su sljedeći:

1. **supposed_url**: Link koji pauk prati sa sjedišta kataloga.
2. **callback**: Callback funkcija *parse_actual_url*, koja se poziva nakon uspješnog praćenja tog linka.
3. **errback**: Callback funkcija *errback_httpbin*, koja se poziva u slučaju neuspješnog praćenja linka.
4. **meta**: Ovaj argument prosljeđuje *WwwItem* objekt ("item": item) callback funkciji *parse_actual_url*, kako bi se sačuvali i dalje koristili podaci prikupljeni u funkciji *parse_item*.
5. **dont_filter**: Postavljeno na *True*, kako bi se omogućilo praćenje istih linkova više puta, što je korisno u slučaju kada više različitih sjedišta kataloga vode na isti link.

Sljedeće poglavlje obuhvaća scenarij uspješnog praćenja linka sa sjedišta, gdje se poziva *parse_actual_url*. Ako dođe do pogreške i poziva *errback_httpbin*, unutar te funkcije zabilježeni su detalji o pogrešci, uključujući HTTP status kod koji se sprema u polje *status_code* objekta *WwwItem*. Više o tome bit će rečeno u poglavlju *3.3. Rezultati*.

3.2.2. Dodatne funkcionalnosti

Osnovne funkcionalnosti pauka omogućuju prikupljanje podataka sa stranica sjedišta sadržanih u katalogu putem prethodno opisane funkcije *parse_item*. Također je definirano praćenje specifičnog linka (Slika 3.2: Link naznačen plavom bojom) koji se nalazi u HTML-u svake od tih stranica. Dodane su nove callback funkcije *parse_actual_url* i *error_httpbin* koje se mogu pozvati prilikom praćenja tog linka. Poziva se samo jedna od njih ovisno o tome je li link uspješno praćen nije.

Cilj dodatnih funkcionalnosti je formirati skup podataka za treniranje neuronske mreže na temelju podataka prikupljenih sa stranica sjedišta u katalogu i s "prave" stranice.

Unutar *parse_actual_url* callback funkcije pozivaju se dodatne funkcije koje koriste podatke sakupljene paukom za izračun ključnih vrijednosti. One se pohranjuju

u polja definirana u razredu *WwwItem*, koja su prethodno unutar funkcije *parse_item* bila inicijalizirana na neke default vrijednosti.

Takve vrijednosti predstavljaju značajke uzoraka skupa podataka koji će se koristiti kao ulaz u neuronsku mrežu. Svaki uzorak predstavlja jedno sjedište iz kataloga, dok skup takvih uzoraka čini skup podataka koji može biti korišten za treniranje i testiranje neuronske mreže.

Ključne vrijednosti pohranjena su u polja: *url_match*, *html_match*, *contrast_score*, *text_density_score* i *layout_score*. Prve dvije vrijednosti odnose se na usporedbu URL-a i HTML-a sjedišta u katalogu s onima na "pravoj stranici", dok se preostale tri dobivaju analizom slike naslovnice "prave" stranice. Vrijednosti dobivene analizom slike bit će sažete u jednu natuknicu *snapshot_score*. Funkcije navedene u naredne tri natuknice bit će detaljno opisane u poglavlju *4.1. Skup podataka*.

1. ***url_match***

   U polje *url_match* pohranjuje se rezultat funkcije *compare_urls*. Funkcija vraća neku vrijednost realnog broja u rasponu [0, 1].

2. ***html_match***

   U polje *html_match* pohranjuje se rezultat funkcije *compare_description*. Funkcija vraća neku vrijednost realnog broja u rasponu [0, 1>.

3. ***snapshot_score***

   U polja *contrast_score*, *text_density_score* i *layout_score* pohranjuje se rezultat funkcije *get_screenshot_from_file*. Funkcija za svaku od njih vraća vrijednost realnog broja u rasponu [0, 1].

Važno je napomenuti da postoje dvije različite verzije pauka. Razlika između ta dva pauka leži u callback funkciji *parse_actual_url*.

Prvi pauk, koji je opisan do sada (*kataloger.py*), dobiva vrijednosti polja vezanih uz analizu slike pomoću funkcije *get_screenshot_from_file* unutar callback funkcije *parse_actual_url*. Ta funkcija za svako sjedište vraća rezultat na temelju analize slike naslovnice "prave stranice", koja je pohranjena u direktoriju *Snapshots* unutar projekta. Za kompletan obilazak domene *www.hr* trebalo je nešto manje od dva sata.

U drugom pauku (*kataloger_collector.py*) *parse_actual_url* je modificirana tako da umjesto poziva funkcije *get_screenshot_from_file* poziva funkciju *capture_screenshot*. U ovom pauku, slike naslovnica se dobivaju koristeći Selenium.

Stranice svakog sjedišta se tijekom obilaska pauka učitavaju u pregledniku *Firefox* bez grafičkog sučelja (*headless mode*). Slike naslovnica pohranjuju se u formatu "broj_sjedista.png", a njihova rezolucija je smanjena na 300 x 225 piksela. Ako se stranica ne učita u zadanom vremenu od maksimalno 20 sekundi, nakon tri pokušaja, slika naslovnice se neće pohraniti. Ta su ograničenja postavljena kako bi se omogućilo učitavanje stranica koje se sporo učitavaju ili dinamičkih stranica koje koriste JavaScript, a koje također često trebaju više vremena za učitavanje.

Zbog ovakvog pristupa, trajanje obilaska pauka značajno se produžilo. Za cjelokupnu domenu *www.hr*, pauku *kataloger_collector.py* je trebalo 19 sati, što je preko 17 sati dulje u usporedbi s paukom *kataloger.py*. Pauk *kataloger_collector* pokrenut je samo jednom, isključivo radi prikupljanja slika naslovnica, dok su se metode za analizu tih slika testirale u osnovnom pauku *kataloger.py*.

3.3. Rezultati

Pauk *kataloger.py*, nakon uspješnog obilaska, generira CSV datoteku koja sadrži sve podatke potrebne za daljnju analizu. Tijekom jednog obilaska, prikupljene su informacije o 22.712 sjedišta.

Za svako sjedište, tijekom obilaska, u polje *status_code* upisana je vrijednost ovisno o tome je li link sa sjedišta uspješno praćen ili nije. Ako je link uspješno praćen, tj. parsiran funkcijom *parse_actual_url*, zabilježen je HTTP status kod 200. U suprotnom, funkcija *errback_httpbin* bilježi odgovarajući HTTP status kod ili drugu vrstu pogreške koja može nastati prilikom pokušaja učitavanja stranice.

Sjedišta koja u polju *status_code* imaju vrijednost različitu od 200 neće biti uzeta u obzir u daljnjoj analizi kao kandidati za uzorak skupa podataka za treniranje neuronske mreže. Naime, bez uspješnog praćenja linka, nemoguće je dobiti vrijednosti za polja *html_match*, *contrast_score*, *text_density_score* i *layout_score*, koja su ključna jer čine četiri od pet značajki uzorka. U tablici 3.1 prikazani su najčešće dobiveni dobiveni HTTP status kodovi i druge pogreške pri učitavanju, uz pripadajući broj stranica.

Tablica 3.1: Pregled HTTP status kodova i broja odgovarajućih sjedišta prikupljenih tijekom obilaska pauka

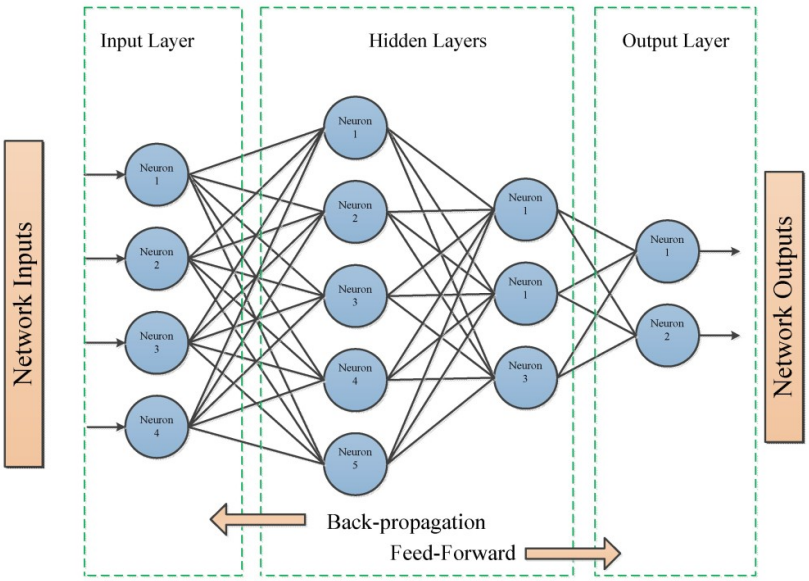
status_code	Broj sjedišta
200 (OK)	16384
401 (Unauthorized)	36
403 (Forbidden)	241
404 (Not Found)	460
410 (Gone)	23
500 (Internal Server Error)	140
503 (Service Unavailable)	60
DNS Lookup Failed	4463
Timeout	801
Unknown Error	72
Ostali HTTP kodovi	32

Većina sjedišta koja su dobila status kod različit od 200 može se smatrati neaktivnima, odnosno u kontekstu zadatka lažnima. Međutim, neke stranice prepoznaju da ih posjećuje pauk te mu onemogućuju pristup, što rezultira status kodom 403 (Forbidden). Taj problem nastaje kada web stranice detektiraju da je zahtjev poslan od strane pauka, a ne stvarnog korisnika. To se u većini slučajeva može zaobići promjenom atributa *USER_AGENT* u datoteci *settings.py* koja se nalazi u *Scrapy* projektu.

Iznimno postoji 349 sjedišta sa HTTP status kodom 200 za koje polja *contrast_score*, *text_density_score* i *layout_score* nisu zadobila valjanu vrijednost jer pauk *kataloger_collector.py* za ta sjedišta nije uspio dobiti sliku naslovnice. Razlog tome je predugo učitavanje tih sjedišta prilikom pokušaja spremanja slike naslovnice. Takva se sjedišta također neće razmatrati u daljnjoj analizi.

4. Neuronska mreža za detekciju lažnih sjedišta weba

Kako bi postigli rezultate analize web sjedišta u radu je korištena feed-forward neuronska mreža (Slika 4.1). To je osnovni tip umjetnih neuronskih mreža, a taj tip mreže obilježava to da informacije teku u jednom smjeru, od ulaznih prema izlaznim slojevima mreže bez ikakvih povratnih veza. Mreža se sastoji od proizvoljnog broja slojeva neurona. Svaka takva mreža sadrži jedan ulazni i izlazni sloj te proizvoljan broj skrivenih slojeva između njih. Svaki neuron iz jednog sloja povezan je sa svakim neuronom sljedećeg sloja, a težine tih veza se prilagođavaju kroz proces treniranja mreže putem *backpropagation* algoritma (algoritam propagacije pogreške unazad)[4].



Slika 4.1: Osnovna struktura feed-forward neuronske mreže (FNN) [17]

Mreža koristi nadzirano učenje, što znači da mreža za treniranje koristi unaprijed označene podatke. Cilj modela tako trenirane mreže je na temelju ulaznih podataka i njima zadanih oznaka (ciljnim vrijednostima), generalizirati pravila koji se kasnije mogu primijeniti na nove, nepoznate primjere. U ovom radu razmatra se rješenje binarnom klasifikacijom gdje skup podataka ima dvije moguće oznake.

Skup podataka je ključan u strojnom učenju jer kvaliteta i kvantiteta skupa podataka izravno utječu na učinkovitost modela. Skup podataka za učenje ove mreže bit će detaljnije opisan u narednom poglavlju.

4.1. Skup podataka za učenje neuronske mreže

Skup podataka korišten za učenje mreže sastoji se od uzoraka oblika:

url_match, html_match, contrast_score, text_density_score, layout_score, label

Prvih pet vrijednosti u tom uzorku predstavljaju značajke (engl. *features*) koje opisuju jedno sjedište, dok posljednja vrijednost predstavlja pripadajuću oznaku (engl. *label*). Oznaka može biti "0" za ispravno sjedište i "1" za lažno sjedište.

Tih pet značajki je za svako sjedište prikupljeno iz CSV datoteke generirane paukom *kataloger.py* nakon obilaska cijelog kataloga. Skup podataka sastavljen je ručnim odabirom sjedišta iz te datoteke te naknadnim pridodavanjem oznake "0" ili "1" ovisno o tome je li sjedište procijenjeno kao lažno ili nije. U svrhu treniranja prikupljeno je 500 takvih uzoraka od kojih je 250 označeno s "0", a 250 s "1".

Razlog ravnomjernoj raspodjeli tih oznaka, je potreba za jednakom reprezentacijom obje klase ("0" i "1") unutar skupa podataka. Naime u manjim skupovima podataka kao što je ovaj, takvim se pristupom mreži omogućava da podjednako "uči" na primjerima obje klase. Time se umanjuje rizik pristranosti jednoj od te dvije klase. Takvom raspodjelom trebali bi dobiti precizniji model s koji bolje generalizira[15].

Uz skup za treniranje potrebno je prikupiti i skup za testiranje nad kojem testiramo dobiveni model. Skup za testiranje sastoji se od 100 uzoraka, također s ravnomjernom raspodjelom oznaka.

Značajke za skup podataka odabrane su analizom mogućih rješenja za detekciju lažnih sjedišta. U svojoj ulozi mogu se podijeliti u dvije kategorije:

1. **Značajke za analizu slika sjedišta**: *contrast_score*, *text_density_score*, *layout_score*
2. **Značajke za analizu sadržaja i vlasništva sjedišta**: *url_match*, *html_match*

4.1.1. Značajke za analizu slika sjedišta

Slika 4.2: Primjer slike u sivim tonovima (engl. *grayscale image*) [3]

Značajke za analizu sjedišta pružaju nam informacije o potencijalnim anomalijama na slici naslovnice sjedišta[8]. Takvom se analizom nastoji detektirati sjedišta koja su neaktivna ili žrtva defacementa. Analiza slika se izvodi unutar pauka učitavanjem slika sjedišta umanjenih na veličinu 300 x 225 piksela.

```
gray_img = cv2.cvtColor(img_np, cv2.COLOR_BGR2GRAY)
contrast = np.std(gray_img)realnih brojeva)
```

Prva takva značajka je ***contrast_score***. Slika naslovnice koja se analizira prvo se pretvara u sliku sa sivim tonovima (engl. *grayscale image*) (Slika 4.2). Slika sa sivim tonovima dobiva se pretvorbom iz originalne slike te u njoj svaki piksel ima vrijednost između 0 (crna boja) i 255 (bijela boja)[10]. Standardna devijacija mjeri koliko intenzitet svakog piksela odstupa od prosječne vrijednosti intenziteta u slici. Što je devijacija veća, to je veći kontrast jer su veće razlike između svjetlijih i tamnijih dijelova slike.

Kako bi normalizirali tu vrijednost na raspon [0, 1], standardna devijacija se dijeli s maksimalnom vrijednošću intenziteta piksela, koja je 255. Tako dobivamo *contrast_score* značajku, odnosno relativnu mjeru kontrasta koja je bazirana na razlikama između piksela, a ne na vrijednostima zasebnih piksela.

Sjedišta koja su žrtva defacementa često imaju veoma visoku vrijednost kontrasta, dok neaktivne stranice, koje su nerijetko skoro bez sadržaja i jednobojne, imaju vrlo nisku vrijednost kontrasta.

Sljedeća značajka vezana uz analizu slike sjedišta je ***text_density_score***, odnosno gustoća teksta. Prvo prethodno dobivenu sliku u sivim tonovima pretvaramo u binarnu sliku (engl. *binary image*). Svaki piksel koji ima vrijednost manje od 128 (tamniji) pretvaramo u 255 (bijelu boju), a one veće od 128 (svjetliji) u 0 (crnu boju). Tako dobiveni bijeli pikseli označuju tekst, a proporcija tih piksela naspram ukupne površine slike daje vrijednost u rasponu [0, 1] značajki *text_density_score*.

```
_, binary_img = cv2.threshold(gray_img, 128, 255,
                              cv2.THRESH_BINARY_INV)
text_density = np.sum(binary_img) / 255
text_density /= (binary_img.shape[0]
                 * binary_img.shape[1])
```

Veća vrijednost nam govori da na slici postoji veća količina potencijalnog teksta (mogu biti i tamne površine). S druge strane manja vrijednost ukazuje na manju količinu potencijalnog teksta. Niska vrijednost takve značajke može ukazivati na neaktivne stranice koje često sadrže malenu količinu teksta.

Zadnja značajka vezana uz analizu slike sjedišta je ***layout_score***. Tom značajkom se analizira raspored elemenata na stranici sjedišta. Elementi mogu biti dijelovi teksta, slike, reklame itd. Kako bi dobili tu vrijednost također je potrebna binarna slika opisana kod značajke *text_density_score*. Elementi, odnosno povezane komponente (engl. *Connected Components*)[9] se na binarnoj slici detektiraju kao povezana skupina bijelih piksela dok se crni pikseli uzimaju kao pozadina koja se ne razmatra.

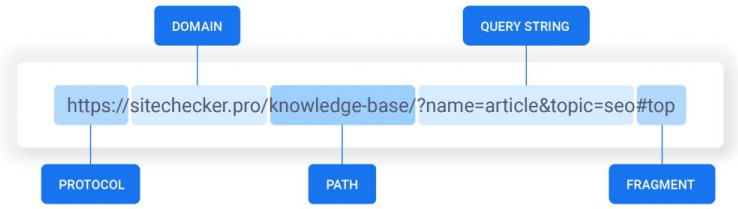
```
num_labels, labels_img = cv2.connectedComponents(
                                binary_img)
layout_score = 1.0 / min(num_labels, MAX_LABELS) \
                    if num_labels > 0 else 0
```

Manji broj takvih elemenata može ukazivati na jednostavnije stranice, dok veći broj elemenata često označava složenije stranice s razvijenijim rasporedom. Kao i preostale značajke vezane uz analizu slike, *layout_score* se normalizira na vrijednost u rasponu [0, 1]. Stranice s manjim brojem elementa dobit će veći *layout_score*, a one s više elemenata manji *layout_score*. Neaktivne stranice često imaju manji broj elemenata pa će dobiti viši *layout_score*. Velik broj defacement napada izvodi se tako što se preko velikog dijela naslovne stranice dodaje nekakva reklama, koja će se u ovom slučaju smatrati jednim elementom. Takve bi stranice trebale zadobiti viši *layout_score*.

4.1.2. Značajke za analizu sadržaja i vlasništva sjedišta

Značajke za analizu sadržaja i vlasništva služe kako bi dodatno proširila funkcionalnost analize slike sjedišta tako što uvidom u HTML i URL sjedišta mogu procijeniti odgovara li sadržaj i vlasništvo onome navedenom u katalogu. Te dvije značajke su *url_match* i *html_match*. Njima se vrijednosti dodjeljuju tijekom obilaska pauka.

Prva od tih značajki je ***url_match***. Funkcija za izračun te značajke implementira sustav bodovanja sličnosti URL-a navedenog u katalogu i pravog URL-a sjedišta. Bodovanje kreće od početne vrijednosti na koju se dodaje bonus u slučaju sličnosti i kazna u slučaju razlike. Ako su URL-ovi identični *url_match* dobiva vrijednost 1, a u slučaju da bodovanjem nije uspostavljen nikakav bonus, dobiva vrijednost 0. U suprotnom poprima vrijednost nekog realnog broja u rasponu [0, 1].



Slika 4.3: Struktura URL-a [1]

Usporedba URL-ova napravljena je na principu strukture samog URL-a. Zasebno se boduje sličnost domene, putanje i query-a. Razlika u protokolu se ne razmatra pri bodovanju. Dakle ako je jedina razlika pri usporedbi dva URL-a protokol (npr. http i https), URL-ovi se smatraju identičnima.

Bodovanje osim sličnosti u osnovnim elementima URL-ova također u obzir uzima ekstenzije koje se mogu naći na kraju putanja URL-a sjedišta kao i neke ključne riječi unutar istog URL-a. Ako putanja URL-a sjedišta sadrži određene ekstenzije, dodaje se malen bonus ako se preklapaju s ekstenzijama u definiranoj listi ekstenzija (npr. ".html", ".php", ".php/hr/").

Isti princip vrijedi za ključne riječi (npr. "/hr/", "/home"). Jedina razlika je u tome što za ključne riječi postoji i lista koja sadrži nepoželjne ključne riječi (npr. "404", "error", "notfound"). Pojavom njih u URL-u sjedišta, ukupnoj vrijednosti bodova se dodaje kazna. Izvedba bodovanja detaljno je opisana komentarima unutar koda.

Druga korištena značajka je ***html_match***. Funkcija za izračun te značajke implementira sustav za bodovanje sličnosti sadržaja HTML-a sjedišta i njegova opisa u katalogu. Na temelju rezultata bodovanja, značajka *html_match* poprima vrijednost nekog realnog broja u rasponu [0, 1>.

Katalog može sadržavati kratki opis sjedišta te naslov sjedišta. Na slici 3.2 vidimo jednu takvu stranicu kataloga, gdje je opis sjedišta tekst neposredno ispod "*OPIS*", a naslov "*About Croatia*".

Prvi korak funkcije koja vraća vrijednost za *html_match* je usporedba naslova iz kataloga s naslovom sjedišta. Naslov sjedišta dobiva se CSS selektorom koji unutar HTML-a sjedišta traži element <*title*>. Ukoliko se naslov pronađe, uspoređuje se s onim iz kataloga te se na temelju sličnosti dodaju bonusi a na temelju razlika kazne. Kod naslova je koristeći Python biblioteku *Levenshtein* omogućen dodatan mehanizam usporedbe naslova, odnosno bodovanja[7]. U slučaju da se naslov ne pronađe, taj dio bodovanja neće se izvršiti.

Sljedeći korak funkcije za bodovanje je usporedba sadržaja HTML-a sjedišta s ključnim riječima opisa i naslova iz kataloga. Za ključne riječi su uzimaju skoro sve one koje se pojavljuju unutar opisa i naslova kataloga. Iznimno se ne koriste one koje su kraće od tri slova ili neke iz definirane liste *COMMON_WORDS* (npr. "the", "and", "site", "com"). Za svaku pronađenu ključnu riječ unutar HTML-a sjedišta dodaje se bonus, a za svaku ne-pronađenu kazna. Dodana je limitacija da se pojava jedne ključne riječi može zabilježiti najviše pet puta. Izvedba bodovanja detaljno je opisana komentarima unutar koda.

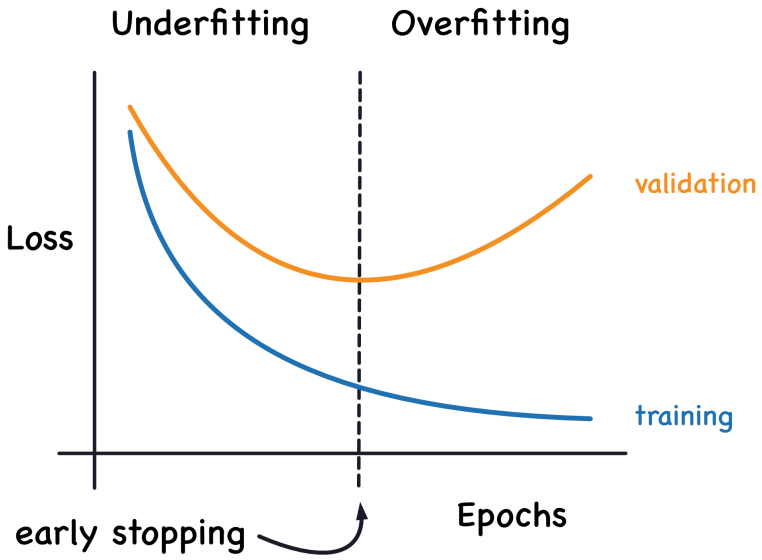
4.2. Implementacija mreže

Neuronska mreža, trenirana na skupu podataka opisanom u prethodnom poglavlju, implementirana je u programskom jeziku *Python* unutar radnog okvira *TensorFlow*. *TensorFlow* pruža fleksibilne i pristupačne opcije za implementaciju neuronskih mreža.

Tijekom treniranja mreže testirane su tri različite arhitekture. Sve ove mreže dijele zajedničke karakteristike: ulazni sloj sadrži pet neurona, pri čemu svaki neuron predstavlja jednu značajku iz skupa podataka, dok izlazni sloj sadrži jedan neuron. Neuron na izlaznom sloju koristi sigmoidalnu aktivacijsku funkciju za predviđanje oznaka ("0" ili "1"), što je potrebno za binarnu klasifikaciju.

Razlika između navedenih arhitektura leži u broju skrivenih slojeva i broju neurona unutar svakog sloja. Prva mreža sastoji se od tri skrivena sloja koji redom imaju 64, 32

i 16 neurona. Preostale dvije mreže imaju dva skrivena sloja. Prva od njih sadrži 32 i 16 neurona, a druga 16 i 8 neurona.



Slika 4.4: Optimalan trenurak prekida učenja [16]

Što ima više skrivenih slojeva ili neurona, raste složenost neuronske mreže, što nije nužno dobro budući da se u sklopu rada ne raspolaže velikim skupom podataka. Treniranjem takve složene mreže s manjim skupom podataka lako može dovesti do pretreniranosti mreže (engl. *overfitting*). Pretreniran model odlično će predviđati na skupu za treniranje, no na neviđenom skupu će loše generalizirati.

Suprotno tome, ako mreža ima premalo skrivenih slojeva ili neurona, smanjuje se složenost neuronske mreže, a time mreža gubi mogućnost generaliziranja na skupu za treniranje (engl. *underfitting*), a time i tako dobiven model nema dobru sposobnost generalizacije na neviđenom skupu podataka.

Prilikom treniranja mreža s dva skrivena sloja uočen je *underfitting*, dok je mreža s tri sloja davala dobre početne rezultate na skupu za treniranje. U daljnjoj analizi koristi se takva mreža s tri skrivena sloja.

Kako bi trenirali mrežu potrebno je podijeliti skup podataka na skup za treniranje i skup za validaciju. Skupu za treniranje daje se 400 uzoraka podataka, a skupu za validaciju 100. Prilikom toga pazi se na raspodjelu oznaka ("0" ili "1") kako bi se u treniranju obje klase jednako reprezentirale. Stoga će u skupu za treniranje biti 200 uzoraka s oznakom "1" te 200 s oznakom "0". Takva ravnomjerna raspodjela bit će primijenjena i na skup za validaciju (50/50).

Proces treniranja odvija se u epohama. Prije početka prve epohe težine se postavljaju na početne vrijednosti. Jedna epoha definira prolazak kroz cijeli skup podataka za treniranje. Tijekom jedne epohe treniranje se provodi u manjim grupama uzoraka podataka odnosno *batch*-evima. Hiperparametrom *batch_size* zadajemo koliko će uzoraka podataka model koristiti prije nego ažurira svoje težine. Npr. ako je *batch_size* postavljen na 10, a skup za treniranje sadrži 400 uzoraka podataka, potrebno je 40 takvih grupa da se završi jedna epoha. Odabir hiperparametra *batch_size* veoma je bitan za treniranje modela. Naime, ako je *batch_size* premalen, treniranje može biti suboptimalno i sporo zbog češćih ažuriranja težina. To može dovesti do nestabilne konvergencije modela. Ako je *batch_size* prevelik, model može postati skloniji pretreniranju jer uči u prevelikim koracima i može zaglaviti u lokalnim minimumima, što rezultira slabijom sposobnošću generalizacije.

Hiperparametri koji značajno utječu na proces treniranja su stopa učenja, te odabrani optimizator i funkcija gubitka. Stopa učenja (engl. *learning rate*) određuje veličinu koraka koje model uzima prilikom ažuriranja težina. Ako je stopa učenja preniska, model će se trenirati vrlo sporo i sklon je zaglavljivanju u lokalnim minimumima. Previsoka stopa učenja može onemogućiti konvergenciju modela, odnosno uskratiti mu sposobnost generalizacije (*underfitting*). *TensorFlow* nam nudi opciju korištenja *ADAM* optimizatora koji prilagođava stope učenja pojedinačno za svaku težinu. To modelu omogućuje stabilniju i bržu konvergenciju. Za funkciju gubitka koristi se funkcija "*binary cross-entropy*". Tom funkcijom mjerimo razliku između stvarnih oznaka uzoraka podataka i modelom predviđenih vrijednosti. Kako bi se postiglo što bolje generaliziranje modela, cilj je minimizirati taj gubitak.

Nakon svake epohe, sposobnost generalizacije modela evaluira se na validacijskom skupu. Evaluacijom dobivamo validacijski gubitak (engl. *validation loss*) i točnost (engl. *validation accuracy*). Cilj je treniranje modela zaustaviti u onom trenutku kada gubitak na skupu za validaciju prestane padati i krene rasti (Slika 4.4). Takvo ponašanje implementirano je metodom ranog zaustavljanja (engl. *early stopping*). Parametar koji definira ponašanje ranog zaustavljanja je *patience*. On nam govori koliko epoha može proći bez poboljšanja validacijskog gubitka prije nego se treniranje zaustavi. Model se zatim vraća na najbolje spremljene težine iz epohe u kojem je zabilježen najniži validacijski gubitak. Time osiguravamo optimalnu učinkovitost modela.

Kako bi što više optimirali sposobnost generalizacije naše mreže, potrebno je podešavati prethodno navedene hiperparametre, odnosno sve one parametre koji se kroz eksperimentaciju podešavaju u svrhu poboljšanja učinkovitosti modela. Treniranjem osnovnog modela ustanovljene su vrijednosti nekih od hiperparametara koje će se koristiti u daljnjoj analizi rješenja. Neki od tih hiperparametara i njima pridružene vrijednosti zadane su u tablici.4.1

Tablica 4.1: Vrijednosti hiperparametara

Hiperparametar	Vrijednost
Learning rate	0.0005
Batch size	8
Patience	50

Budući da je skup podataka malen, sama optimizacija hiperparametara nije nužno dovoljna za adekvatno poboljšanje učinkovitosti modela jer je model i dalje sklon pretreniranju. Kako bi se potencijalno poboljšala učinkovitost modela, na raspolaganju imamo metode regularizacije koje pomažu u smanjenju složenosti modela i poboljšavaju njegovu sposobnost generalizacije na novim, neviđenim uzorcima[19].

Uz već objašnjenu metodu ranog zaustavljanja, razmatrane su i sljedeće metode regularizacije:

1. **L2 regularizacija**
2. **Dropout**
3. **L2 + Dropout**

L2 regulacija[11] dodaje dodatnu kaznu težinama u modelu tako što modificira funkciju gubitka. Prije ažuriranja svake težine na funkciju gubitka dodaje se suma kvadrata svih težina neurona, pomnožena s hiperparametrom regularizacije (*L2_value*). Time dobivamo model koji preferira manje težine, ali istovremeno informativne težine. Kod većih težina, model je skloniji prilagođavanju njima, što uzrokuje pretreniranost modela.

Veoma je bitno paziti na odabir tog hiperparametra. Kod odabira prevelike vrijednosti, težine se previše smanjuju i model gubi mogućnost generalizacije. U suprotnom, ako je vrijednost hiperparametra premalena, L2 regularizacija neće znatno utjecati na smanjivanje pretreniranja. Stoga se preporučuje eksperimentalno određivanje optimalne vrijednosti L2 regularizacije uz podešavanje ostalih parametara.

Tijekom treniranja modela isprobane su različite vrijednosti za hiperparametar L2 regularizacije te je balans između prethodno opisanog ponašanja postignut vrijednošću 0.0001. Ostali hiperparametri u sklopu treniranja takvog modela pokazali su se optimalnima u vrijednostima koje su prikazane na tablici 4.1.

Dropout[12] regularizacija nasumično isključuje određen postotak neurona, tj. težine im postavlja na nulu i ne ažurira ih tijekom jedne epohe. Taj postotak definira se hiperparametrom stope dropouta *dropout_rate*. To radimo kako bi model postao manje ovisan o nekim specifičnim neuronima. Time potičemo model na bolje korištenje značajki iz podataka, što utječe na smanjenje pretreniranosti.

Za hiperparametar *dropout_rate* isprobane su četiri vrijednosti: 0.1, 0.2, 0.3 i 0.4. Već kod vrijednosti 0.2 moglo se uočiti da model gubi sposobnost generalizacije, a daljnjim povećavanjem vrijednosti taj je efekt bio sve više izražen. Stoga je za daljnje testiranje hiperparametra *dropout_rate* izabrana vrijednost 0.1, što znači da se prilikom treniranja u svakoj epohi nasumično isključi 10% neurona. Uz novi hiperparametar *dropout_rate* kod treniranja. Ostali hiperparametri u sklopu treniranja takvog modela pokazali su se optimalnima u vrijednostima koje su prikazane na tablici 4.1.

Istovremenom primjenom **L2** i **dropout** regularizacije pri treniranju modela kombiniraju se njihove prednosti. Naknadnim treniranjem takvog modela i testiranjem različitih vrijednosti hiperparametara, uspostavljene su sljedeće optimalne vrijednosti hiperparametara tih dviju regulacija: *L2_value = 0.0001*, *dropout_rate* = 0.1. Ostali hiperparametri u sklopu treniranja takvog modela pokazali su se optimalnima u vrijednostima koje su prikazane na tablici 4.1.

4.3. Rezultati evaluacije modela

U svrhu dobivanja rezultata trenirana su četiri različita modela. Svaki model treniran je koristeći regularizaciju ranog zaustavljanja (engl. *early stopping*). Razlika između tih modela je u dodatnim metodama regularizacije koje koriste. Nazivi tih modela s pripadajućim metodama regularizacije opisani su u tablici 4.2.

Tablica 4.2: Modeli

Naziv modela	Regularizacija
basic_model	Rano zaustavljanje
basic_model_l2	L2 Regularizacija
basic_model_dropout	Dropout
basic_model_l2_dropout	L2 + Dropout

Svaki od tih modela evaluiran je na dosad neviđenom skupu za testiranje. Skup za testiranje sadrži se od 100 uzoraka, od kojih 50 ima oznaku "1" (lažno sjedište), a 50 oznaku "0" (ne lažno sjedište). Modeli daju predikcije na testnom skupu. Ako je vrijednost na izlazu modela veća od 0.5, model će tom uzorku pridijeliti oznaku "1", a u suprotnom oznaku "0".

Za usporedbu performansi modela koristi se **F1 mjera** (engl. *F1 score*)[5]. Pomoću F1 mjere mjerimo preciznost modela na skupu podataka za testiranje. F1 mjera, prikazana formulom 4.3, predstavlja harmoničku sredinu između preciznosti (engl. *precision*) i odziva (engl. *recall*).

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{4.1}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4.2}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.3}$$

Preciznost (engl. *Precision*), prikazana formulom 4.1, mjeri koliko je točnih pozitivnih (engl. *True Positives*) predikcija napravljeno od ukupnog broja pozitivnih predikcija, odnosno zbroja pravih pozitivnih i lažnih pozitivnih predikcija (engl. *False Negatives*).

Odziv (engl. *Recall*), prikazan formulom 4.2, mjeri koliko je stvarnih pozitivnih slučajeva točno predviđeno od ukupnog broja stvarnih pozitivnih slučajeva, odnosno zbroja pravih pozitivnih i lažnih negativnih (engl. *False Negatives*).

Svrha F1 mjere je da uravnoteži lažne pozitivne i lažne negativne predikcije i prikaže kako model balansira preciznost i odziv.

Uz F1 mjeru, performanse modela predstavljaju i gubitak te točnost na skupu za testiranje. Ako je gubitak na skupu za testiranje (engl. *test loss*) veći od gubitka na skupu za validaciju (engl. *validation loss*), model je pretreniran. Drugim riječima,

model na testnom skupu ne generalizira jednako dobro kao što je to predviđeno tijekom treniranja. Što je manja razlika između te dvije vrijednosti, to model bolje generalizira na novom skupu za testiranje.

Tablica 4.3: Rezultati evaluacije modela

Model	F1 Score	Test Loss	Validation Loss	TL - VL
basic_model	0.7451	0.5793	0.4071	0.1722
basic_model_l2	0.72	0.6145	0.4095	0.205
basic_model_dropout	0.707	0.5761	0.4390	0.1371
basic_model_l2_dropout	0.7647	0.5511	0.4224	0.1287

Na temelju informacija iz tablice 4.3 možemo analizirati performanse modela. Kod svih modela uočava se pretreniranost, no u nekima je ona više izražena.

Zasebno korištenje L2 i dropout regularizacije u modelima *basic_model_l2* i *basic_model_dropout* nije dalo bolje rezultate u odnosu na osnovni model *basic_model*. Kao što je prikazano rezultatima evaluacije modela *basic_model_l2*, L2 regularizacija se nije pokazala kao dobra metoda regularizacije za dan skup podataka. S druge strane, model *basic_model_dropout* ima bolje svojstvo generalizacije od osnovnog modela, ali pod cijenu lošije F1 mjere. To je posljedica korištenja dropout regularizacije. Iako on smanjuje pretreniranost modela, smanjenjem broja aktivnih neurona, takav model ne uči dovoljno značajki. Što je manji skup podataka, taj će takav efekt biti izraženiji.

Model koji se pokazao najuspješnijim je, suprotno očekivanjima, onaj koji kombinira dropout i L2 regularizaciju (*basic_model_l2_dropout*). Znatno je bolji od modela koji te regularizacije primjenjuju zasebno, a u usporedbi s osnovnim modelom nudi nešto bolju F1 mjeru i bolje svojstvo generalizacije.

Razlog tome je balansiranje tih dviju metoda. Dok L2 regularizacija smanjuje težine modela, dropout sprječava ovisnost o specifičnim neuronima. Tako model može bolje učiti na količinski vrlo ograničenom skupu podataka.

Za 100 uzoraka skupu za treniranje model *basic_model_l2_dropout* je imao 24 krive predikcije.

(a) [26]

(b) [28]

Slika 4.5: Slike naslovnica lažno negativnih sjedišta

Za 50 uzoraka tog skupa, označenih kao lažna sjedišta (oznaka "1"), model je krivo predvidio 11 sjedišta kao ispravna (dao im oznaku "0"), odnosno bilo je 11 lažnih negativnih predikcija. Na slikama 4.5a i 4.5b prikazane su slike naslovnica lažno negativnih sjedišta. Naime oba su sjedišta neaktivna. U oba primjera su značajke respektivnih uzoraka, vezane uz usporedbu HTML strukture, ukazivale na to da je stranica ispravna. Značajke vezane uz analizu slike nisu imale odlučan faktor u predikciji pa je došlo do krive predikcije.

(a) [29]

(b) [30]

Slika 4.6: Slike naslovnica lažno pozitivnih sjedišta

Za 50 uzoraka tog skupa, označenih kao ispravna sjedišta (oznaka "0"), model je krivo predvidio 13 sjedišta kao lažna (dao im oznaku "1"), odnosno bilo je 13 lažnih pozitivnih predikcija. Na slikama 4.6a i 4.6b prikazane su slike naslovnica lažno pozitivnih sjedišta. U oba primjera su značajke respektivnih uzoraka, vezane uz HTML, ukazivale na to da su sjedišta ispravna. Međutim, značajke uzoraka vezane uz analizu slike, koje su neispravno ukazivale da su sjedišta lažna, uzrokovale su krivu predik-

ciju. Također je bitno napomenuti da se slika naslovnice (4.6b) nije uspjela do kraja učitati tijekom sakupljanja slika od strane pauka *kataloger_collector.py* pa značajke vezane uz analizu slike nisu uspjele dobiti vrijednosti koje reprezentiraju stvarnu sliku naslovnice, što je negativno utjecalo na predikciju modela.

Unatoč tome što je model *basic_model_l2_dropout* pokazao bolje performanse u usporedbi s ostalim modelima, i dalje je sklon pretreniranosti. Jedan od razloga za to je ograničena veličina skupa podataka za treniranje. Znatnim povećanjem skupa podataka za treniranje te daljnjim eksperimentiranjem s hiperparametrima modela i metodama regularizacije, mogli bismo dodatno poboljšati performanse modela.

Osim kvantitete skupa podataka, kvaliteta skupa podataka također ima prostora za poboljšanje. Kompleksnijom implementacijom analize slike i HTML strukture dobile bi se bolje značajke uzoraka skupa podataka, čime bi se dodatno ojačala sposobnost modela za generalizaciju.

5. Zaključak

Svrha rada bila je ustanoviti metode analize lažnih sjedišta s naglaskom na analizu slike sjedišta. Kako bi dobili skup podataka koji koristimo za daljnju analizu korištena su sjedišta kataloga WWW.HR.

Prvo se web paukom sakupljaju podaci o sjedištima. Već tada možemo analizom koju obavi pauk dobiti informacije o velikoj količini lažnih sjedišta. Preostala sjedišta daju se na daljnju analizu feed-forward neuronskoj mreži.

Za nadzirano treniranje takve mreže bilo je potrebno ručno sakupiti skup podataka i dodijeliti svakom uzorku skupa podataka pripadajuću oznaku. Postoje dvije oznake, "1" za lažno sjedište i "0" ispravno sjedište. Sakupljeno je 500 uzoraka za treniranje mreže i 100 uzoraka za evaluaciju.

Svaki uzorak, osim oznake, sadrži i pet značajki na temelju kojih neuronska mreža uči razlikovati lažno od ispravnog sjedišta. Prve dvije su *url_match* i *html_match*. Te se značajke dobivaju tijekom paukova obilaska, a izračunate su na temelju analize HTML strukture sjedišta kataloga i sjedišta opisanog u katalogu. Ostale tri značajke su *contrast_score*, *text_density_score* i *layout_score*. One su također dobivene i izračunate tijekom paukova obilaska, ali su rezultat analize slika naslovnica sjedišta.

Kombinacijom ovih značajki u jedan uzorak skupa podataka, mreži je omogućeno da pri analizi lažnih sjedišta istovremeno uzima u obzir analizu slike naslovnice sjedišta te HTML strukture sjedišta kataloga i samog sjedišta.

Za daljnje poboljšanje rada bilo bi potrebno proširiti skup podataka kako bi se mreži omogućilo bolje generaliziranje. Uz prošireni skup podataka, neuronska mreža bi se mogla doraditi na temelju novih rezultata treniranja.

LITERATURA

[1] Adrian Cojocariu. Url structure. https://cognitiveseo.com/blog/23628/url-structure/, 2020. Pristupljeno: 10.8.2024.

[2] Crawlbase. What is a web crawler? use cases and examples explained. https://crawlbase.com/blog/what-is-a-web-crawler-use-cases-and-examples/, 2023. Pristupljeno: 5.6.2024.

[3] Geeksforgeeks. What is a grayscale image? https://www.geeksforgeeks.org/differentiate-between-grayscale-and-rgb-images/, 2024. Pristupljeno: 15.8.2024.

[4] Mbali Kalirane. Gradient descent vs. backpropagation: What's the difference? https://www.analyticsvidhya.com/blog/2023/01/gradient-descent-vs-backpropagation-whats-the-difference/, 2023. Pristupljeno: 11.8.2024.

[5] Rohit Kundu. https://www.v7labs.com/blog/f1-score-guide, 2022. Pristupljeno: 25.8.2024.

[6] David Maimon. The coronavirus pandemic moved life online – a surge in website defacing followed. https://www.govtech.com/security/the-coronavirus-pandemic-moved-life-online–a-surge-in-website-defacing-followed.html, 2020. Pristupljeno: 22.5.2024.

[7] Ethan Nam. Understanding the Levenshtein Distance Equation for Beginners. https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0, 2019. Pristupljeno: 10.8.2024.

[8] OpenCV. Changing the contrast and brightness of an image. https://docs.opencv.org/3.4/d3/dc1/tutorial_basic_linear_transform.html, . Pristupljeno: 15.8.2024.

[9] OpenCV. Structural analysis and shape descriptors. https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html, . Pristupljeno: 17.8.2024.

[10] OpenCV. Image thresholding. https://docs.opencv.org/, . Pristupljeno: 16.8.2024.

[11] Neri Van Otten. L1 and l2 regularization explained, when to use them practical how to examples. https://spotintelligence.com/2023/05/26/l1-l2-regularization/, 2023. Pristupljeno: 23.8.2024.

[12] Parthiban. Dropout regularization in deep learning. https://www.analyticsvidhya.com/blog/2022/08/dropout-regularization-in-deep-learning/, 2024. Pristupljeno: 23.8.2024.

[13] PostmanAPI. Postman api platform. Pristupljeno: 24.5.2024.

[14] Patrick Putnam. Script kiddie: Unskilled amateur or dangerous hackers? https://www.uscybersecurity.net/script-kiddie/, 2018. Pristupljeno: 22.5.2024.

[15] Roland L. Dunbrack Jr Qiong Wei. The role of balanced training and testing data sets for binary classifiers in bioinformatics. *PLOS ONE*, 2013. Pristupljeno: 13.8.2024.

[16] Alexis Cook Ryan Holbrook. Overfitting and underfitting. https://www.kaggle.com/code/ryanholbrook/overfitting-and-underfitting, 2023. Pristupljeno: 18.8.2024.

[17] SaffronEdge. Feedforward vs backpropagation ann. https://www.linkedin.com/pulse/feedforward-vs-backpropagation-ann-saffronedge1, 2023. Pristupljeno: 10.8.2024.

[18] Scrapy. Spiders. https://docs.scrapy.org/en/latest/topics/spiders.html. Pristupljeno: 5.6.2024.

[19] Hayden Smith. Overcome the challenges of small data in neural networks. https://www.linkedin.com/pulse/overcome-challenges-small-data-neural-networks-hayden-smith, 2023. Pristupljeno: 20.8.2024.

[20] WaybackMachine. Wayback machine. https://web.archive.org/. Pristupljeno:.

[21] Who.is. Whois search, domain name, website, and ip tools - who.is. https://who.is/. Pristupljeno: 23.5.2024.

[22] WWW.HR. Apartmani ena, otok ugljan. https://www.hr/sjediste/6526, . Pristupljeno: 22.5.2024.

[23] WWW.HR. Virovtica. https://www.hr/sjediste/36760, . Pristupljeno: 22.5.2024.

[24] WWW.HR. Vremenska prognoza. https://www.hr/sjediste/29056, . Pristupljeno: 22.5.2024.

[25] WWW.HR. Katalog www.hr. https://www.hr/katalog, . Pristupljeno: 22.5.2024.

[26] WWW.HR. Bujinkan croatia. https://www.hr/sjediste/10029, . pristupljeno: 25.8.2024.

[27] WWW.HR. About croatia. https://www.hr/sjediste/21555, . pristupljeno: 22.5.2024.

[28] WWW.HR. Radioton webshop. https://www.hr/sjediste/32318, . pristupljeno: 25.8.2024.

[29] WWW.HR. Obiteljsko gospodarstvo saric. http://www.obiteljskogospodarstvosaric.hr/, . pristupljeno: 25.8.2024.

[30] WWW.HR. Hexmat d.o.o. https://hexmat.hr/, . pristupljeno: 25.8.2024.

[31] WWW.HR. Sat hrvatskoga. https://www.hr/sjediste/2475, . Pristupljeno: 22.5.2024.

Detekcija lažnih sjedišta weba analizom slika naslovnica

Sažetak

Ideja ovog rada je korištenjem feed-forward neuronske mreže prepoznati lažna sjedišta u sklopu kataloga WWW.HR. Lažna sjedišta mogu biti one stranice navedene u katalogu koje su vizualno kompromitirane ili je na njima došlo do promjena sadržaja ili vlasnika. U sklopu rješavanja problema koristio se pauk napravljen unutar radnog okvira Scrapy kojim se prvo sakuplja skup podataka nužan za treniranje modela mreže. Nadzirano učenje mreže izvedeno je unutar radnog okvira Tensorflow. Uzorci na kojima se modeli uče pokrivaju aspekte analize slika naslovnica sjedišta kao i analize HTML strukture sjedišta. U radu je evaluirano i uspoređeno četiri različito trenirana modela.

Ključne riječi: detekcija lažnih sjedišta, vizualno kompromitirana sjedišta, WWW.HR katalog, analiza slika, analiza HTML strukture, feed-forward neuronska mreža, web pauk, nadzirano učenje

Detection of fake websites using homepage image analysis

Abstract

The idea of this article is to recognise fake websites within the WWW.HR catalog using a feed-forward neural network. Fake websites may include those listed in the catalog that are visually compromised or have undergone changes in content or ownership. A web scraper built within a Scrapy framework was used to collect the dataset necessary for training the neural network model. Supervised learning of the neural network was conducted within the Tensorflow framework. Training samples cover aspects of analyzing both the website homepage image and HTML structure. The article evaluates and compares four differently trained models.

Keywords: detection of fake websites, visually compromised websites, WWW.HR catalog, image analysis, HTML structure analysis, feed-forward neural network, web scraper, supervised learning