Usluga za automatiziranu procjenu prostorne točnosti skupova podataka opažanja u prirodi

Pavić, Ivor

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 716

USLUGA ZA AUTOMATIZIRANU PROCJENU PROSTORNE TOČNOSTI SKUPOVA PODATAKA OPAŽANJA U PRIRODI

Ivor Pavić

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 716

USLUGA ZA AUTOMATIZIRANU PROCJENU PROSTORNE TOČNOSTI SKUPOVA PODATAKA OPAŽANJA U PRIRODI

Ivor Pavić

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 30. rujna 2024.

DIPLOMSKI ZADATAK br. 716

Pristupnik:	**Ivor Pavić (0036524050)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	izv. prof. dr. sc. Igor Čavrak

Zadatak: **Usluga za automatiziranu procjenu prostorne točnosti skupova podataka opažanja u prirodi**

Opis zadatka:

Kao jedna od posljedica značajne biološke raznolikosti u Hrvatskoj se, od strane različitih dionika, prikuplja velika količina podataka o opažanjima u prirodi. S obzirom na brojnost i raznolikost baza podataka u koje se ta opažanja pohranjuju, nedostatak kvalitetne kontrole, ispravnosti i standardiziranosti zapisa lokacija opažanja predstavlja ozbiljan izazov daljnjem korištenju tih podataka. Postojanje alata za provjeru prostorne točnosti zapisa lokacija, na hrvatskom jeziku, značajno bi doprinijelo kvaliteti prikupljenih skupova podataka, kao i njihovoj učinkovitijoj uporabi, na primjer procjeni distribucije endemičnih i ugroženih vrsta, kao i praćenju širenja invazivnih vrsta. U cilju provjere i osiguranja kvalitete zapisa podataka o opažanjima potrebno je istražiti problematiku procjene prostorne točnosti opisa opažanja događaja u prirodi na temelju tekstnog opisa opažanja i korištenih toponima. Proučiti postojeća rješenja za geoparsiranje tekstnih opisa opažanja na engleskom i hrvatskom jeziku i ocijeniti njihovu uporabivost. Osmisliti i predložiti alat za automatiziranu procjenu prostorne točnosti skupova podataka opažanja u prirodi. Izvesti predloženi alat kao javno dostupnu uslugu, te ocijeniti funkcionalna i nefunkcionalna svojstva na javno dostupnim skupovima podataka opažanja u prirodi dostupnih na hrvatskom jeziku.

Rok za predaju rada: 14. veljače 2025.

Hvala prof. dr. sc. Igoru Čavraku i dr. sc. Filipu Vargi na mentoriranju i pomoći prilikom pisanja ovog rada te obitelji i prijateljima na podršci prilikom cijelog studiranja.

Sadržaj

1. Uvod

Razvoj novih tehnologija i veća dostupnost podataka olakšali su prikupljanje i obradu velikih količina podataka u različitim znanstvenim područjima. Jedno od takvih područja je praćenje biološke raznolikosti, ali i ekoloških promjena pomoću opažanja u prirodi.

Zbog raznolike flore i bogate faune, u Hrvatskoj se prikuplja golema količina podataka o promatranjima biljnih i životinjskih vrsta, njihovoj rasprostranjenosti i promjenama u ekosustavima. Podatci koji se prikupljaju uglavnom sadrže opis lokacije, datum, vrijeme te ključne informacije o samom opažanju (broj jedinki, vrste, ponašanje jedinki i sl.). Općenito, ti podaci dolaze iz različitih izvora—programi zaštite prirode, izvješća građana, znanstvena istraživanja i nevladine organizacije.

Zbog nedostatka standardizacije samog protokola prikupljanja podataka te različitih izvora podataka, dolazi do progrešaka i smanjenja kvalitete podataka. Neke od najčešćih grešaka koje smanjuju kvalitetu podataka su dupli zapisi, nepotpun opis opažanja te pogrešne geografske koordinate. Dupli zapisi i nepotpuni opisi opažanja mogu se primjetiti i ispraviti provjerom podataka od treće strane, no greške nisu toliko lako uočljive kada je riječ o geografskim koordinatama. Postoji mnogo razloga za pogreške u vidu točnosti geografskih koordinata, ali glavne su poteškoće s ručnim unosom i manjak iskustva u korištenju GPS opreme [1].

Automatizirana procjena prostorne točnosti uvelike bi olakšala provjeru te pridonijela većoj točnosti podataka. To bi omogućilo znanstvenicima preciznije praćenje populacije endemskih i ugroženih vrsta, ali i smanjenje rizika koje invazivne vrste predstavljaju za ekosustav.

Kako je većina alata za geoparsiranje[1] optimizirana za engleski jezik, korištenje hrvatskog jezika, morfološki znatno složenijeg, vrlo je izazovno. Naime riječi u hrvatskom jeziku mijenjaju svoj oblik ovisno o gramatičkom kontekstu u kojem se nalaze, što otežava prepoznavanje ispravne lokacije.

Cilj ovog rada je istražiti postojeće metode za geoparsiranje te ih prilagoditi za hrvatski jezik. Također je, osim alata razvijenih specifično za svrhu geoparsiranja, istraženo i korištenje AI modela za dijelove geoparsiranja. U radu su sve razvijene metode testirane na stvarnim podatcima. Dio podataka preuzet je iz znanstvenih radova, dok je dio podataka dobiven opažanjima građana. Rezultati testiranja razvijenih metoda detaljno su opisani i sistematično uspoređeni kako bi se mogli donijeti zaključci o uspjehu pojedine metode. Na kraju su predložena moguća poboljšanja procesa geoparsiranja koja bi dovela do još veće točnosti sustava.

[1]Geoparsiranje je proces pretvaranja dvosmislene reference u strukturirani geografski zapis. Sastoji se od nekoliko koraka, a to su tokenizacija, prepoznavanje imenovanih entiteta, mapiranje imenovanih entiteta na standardizirane forme, geokodiranje (pridruživanje točnih geografskih koordinata prostornim podatcima) te rezolucija prostorne neodređenosti.

2. Korištene tehnologije

U ovom poglavlju dan je sažeti pregled tehnologija i biblioteka korištenih u radu. Svaka tehnologija i biblioteka opisana je u svrhu lakšeg praćenja ostatka rada.

2.1. Docker

Docker je *open-source* platforma za razvoj, upravljanje i puštanje aplikacija u pogon. Sastoji se od sljedećih komponenti: poslužitelj (eng. *Docker daemon*), Docker klijent, Docker registar te Docker objekti (slika, kontejner, volumen).

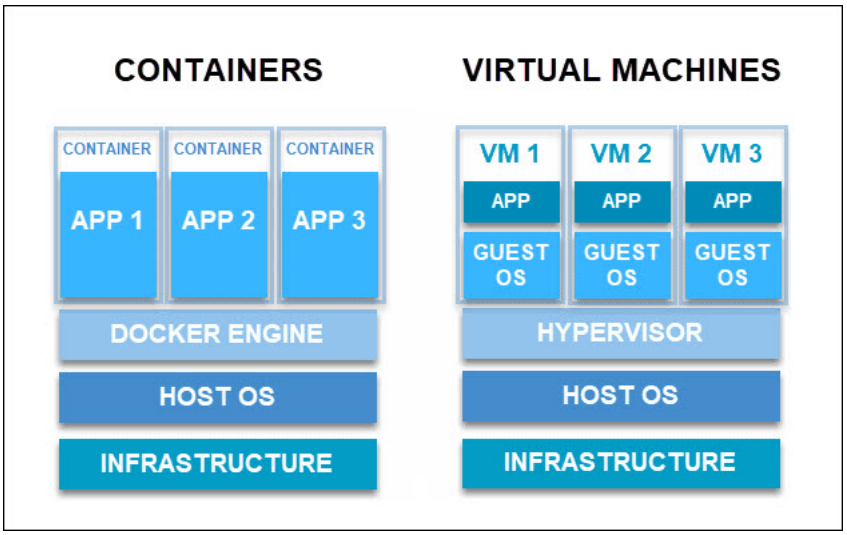
- *Docker daemon* sluša Docker API[1] i upravlja Docker objektima.
- Docker klijent je alat koji korisnicima omogućuje interakciju s *Docker daemonom* na njihovom sustavu. Klijent ima skup Docker CLI[2] naredbi koje omogućuju stvaranje i upravljanje kontejnerima.
- Docker registar pohranjuje Docker slike. Jedan primjer registra je Docker Hub, javni registar koji svatko može koristiti.
- Slika je *read-only* predložak koji služi za kreiranje kontejnera. Nerijetko se slika temelji na nekoj drugoj slici uz dodatne prilagodbe vlastitim potrebama.
- Volumen je trajno spremište podataka za kontejnere, a stvara ga i njime upravlja sam Docker.
- Kontejner je instanca slike koja se može izvoditi. On se može stvoriti, pokrenuti, zaustaviti, premjestiti ili izbrisati koristeći Docker API ili Docker CLI naredbe.

[1] Docker API je programsko sučelje (eng. Application Programming Interface) koje omogućava interakciju s Docker *daemonom* putem HTTP zahtjeva

[2] Docker CLI je alat naredbenog retka koji omogućava korisnicima ručno upravljanje Dockerom putem terminala

Kontejner možemo povezati s jednom ili više mreža, pridružiti mu pohranu ili čak stvoriti novu sliku na temelju trenutnog stanja. Kontejner je definiran svojom slikom kao i bilo kojim konfiguracijskim opcijama koje mu postavimo prilikom stvaranja ili pokretanja. Važno je napomenuti da kada se kontejner ukloni, sve promjene njegovog stanja koje nisu pohranjene u trajnoj pohrani nestaju.

[2]



Slika 2.1. Usporedba dockera i Virtualnog računala, Izvor: [3]

Docker kontjener dakle predstavlja softverski paket s kodom, sistemskim alatima, bibliotekama, ovisnostima i konfiguracijskim datotekama potrebnim za pokretanje određene aplikacije te je svaki kontejner neovisan i izoliran od *hosta* i drugih kontejnera. Ovo uvelike podsjeća na virtualna računala (eng. virtual machines), te iako je koncept sličan, razlike su velike. Kako se može vidjeti na slici 2.1. Docker se izvodi na operacijskom sustavu *hosta* dok svako virtualno računalo mora imati svoj operacijski sustav. Iz tog razloga se veličine Docker kontejnera mjere u megabajtima, dok se veličine virtualnih računala mjere u gigabajtima. Isto tako velika je razlika i u brzini pokretanja; dok se Docker kontejneri pokreću u samo nekoliko sekundi, za virtualna računala to može biti i nekoliko minuta.

2.2. Elasticsearch

Elasticsearch je distribuirani sustav za pretraživanje i analizu podataka u stvarnom vremenu, temeljen na Apache Lucene[3] tražilici. Omogućuje pohranu, pretraživanje i analizu velikih količina podataka. Iako Elasticsearch može obavljati pohranu i dohvat podataka, njegova glavna svrha nije biti baza podataka, već pretraživački sustav (poslužitelj) s glavnim ciljem indeksiranja, pretraživanja i pružanja statistike u stvarnom vremenu na temelju podataka. Elasticsearch ne koristi shemu već se podatci pohranjuju u takozvane JSON objekte zvani dokumenti. Dokumenti sa zajedničkim svojstvima se grupiraju u tipove, a tipovi se zatim pohranjuju u indekse. Za lakšu vizualizaciju može se povući paralela sa relacijskim bazama podataka gdje bi indeks bio baza, tip bi bio neka tablica u bazi, a dokument bi predstavljao jednu n-torku unutar tablice.

Kako bi se ubrzala pretraga, Elasticsearch ima mogućnost distribucije. Odnosno, moguće je pokrenuti više instanci Elasticsearch-a (tzv. čvorovi) te ih rasporediti na različite poslužitelje distribuiranih sustava. Skup čvorova, ili samo jedan čvor ako je jedini, naziva se klaster. Velika sposobnost raspodjele zadataka unutar Elastichsearch klastera uvelike utječe na jako dobre performanse. Također, postoji i mogućnost horizontalne podjele indeksa što se naziva *shard*. *Shard* se zatim gleda kao nezavisan indeks te samim time ubrzava pretragu.

2.3. Spacy

SpaCy je moderna biblioteka za obradu prirodnog jezika u Pythonu dizajnirana za rad s velikim količinama teksta i optimizirana za industrijske primjene. Za razliku od drugih akademski usmjerenih alata i biblioteka, spaCy je izgrađen za učinkovitost i jednostavnost korištenja u stvarnim aplikacijama. Ova biblioteka napisana je u Cythonu[4] te omogućuje izvršavanje naprednih zadataka obrade prirodnog jezika kao što su tokenizacija, označavanje dijela govora, raščlanjivanje ovisnosti i prepoznavanje imenovanih entiteta, za što se i koristi u ovom radu. Nadalje, nudi mogućnost korištenja vektorskih prikaza

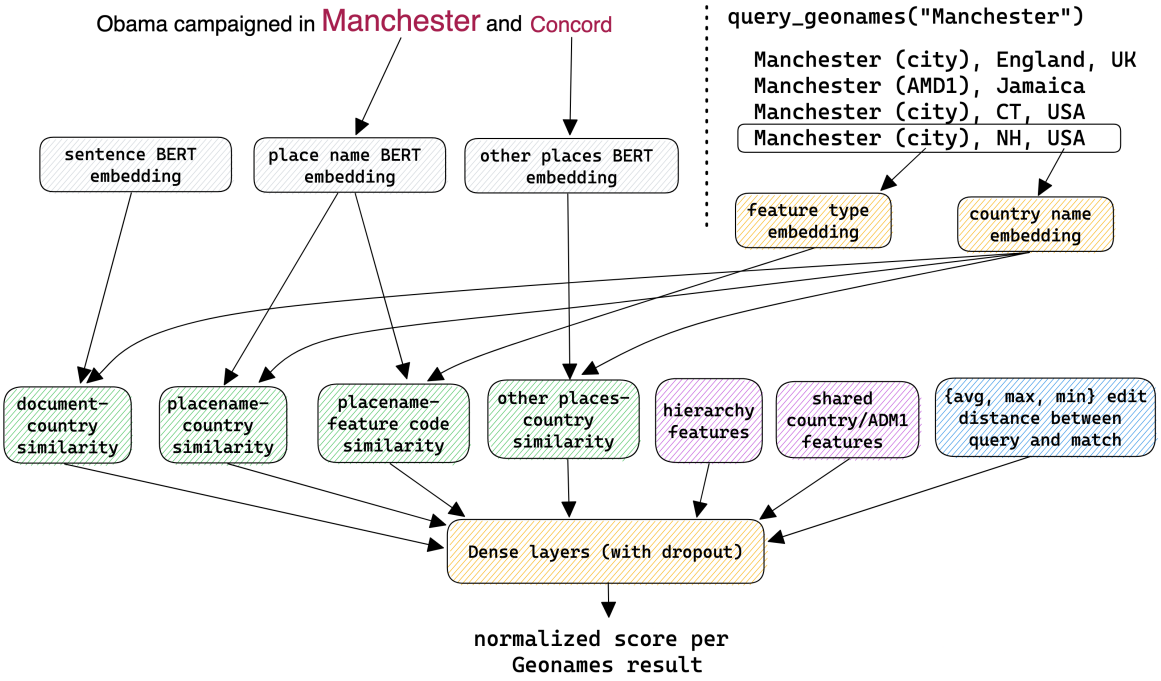
[3] Apache Lucene je *open-source* softver tražilice razvijen u programskom jeziku Java

[4] Ctyhon je proširenje programskog jezika python koji omogućuje kombiniranje programskih jezika C i Python. Osim što omogućuje direktno pozivanje funkcija napisanih u programskom jeziku C, omogućuje i pretvaranje Python koda u C kod radi bržeg izvođenja.

riječi i koristi unaprijed trenirane modele strojnog učenja, što dodatno olakšava analizu semantičkog sadržaja.

2.4. Mordecai3

Mordecai3 (u daljnjem tekstu, radi jednostavnosti, mordecai) je nova verzija postojeće Mordecai biblioteke. Razvijena je u programskom jeziku Python te se koristi za geoparsiranje teksta. Geoparsiranje teksta u ovoj biblioteci provodi se u nekoliko koraka koji su slikovito prikazani na slici 2.2.



Slika 2.2. Slikoviti prikaz geoparsiranja u mordecai biblioteci, Izvor: [4]

Prvi korak u geoparsiranju je prepoznavanje svih lokacija u tekstu. Za to se koristi spaCy biblioteka. Nad tekstom se provodi prepoznavanje imenovanih entiteta, te se uzimaju samo toponimi[5]. Radi veće preciznosti koristi se model koji sadrži transformer[6]. Ovaj dio nije prikazan na slici 2.2. već su lokacije samo istaknute crvenom bojom. Nakon prepoznavanja svih lokacija, mordecai pretražuje Elasticsearch indeks koji sadrži podatke iz GeoNames baze kako bi dobio kandidate za rezultat geoparsiranja. Zatim koristi model neuronske mreže za rangiranje i odabir najboljeg podudaranja među kandi-

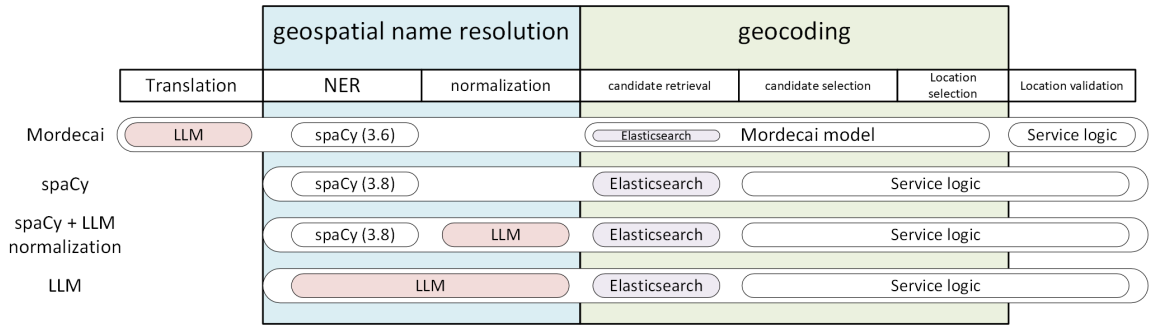
[5]Toponimi su geografska imena, odnosno imena koja se koriste za označavanje specifičnih mjesta na Zemlji, kao što su gradovi, naselja, rijeke, planine, ulice i drugi geografski objekti.

[6]*Transformer* je komponenta koja koristi duboke neuronske mreže za poboljšanje odrađivanja zadataka obrade prirodnog jezika.

datima, oslanjajući se na značajke iz naziva lokacije, drugih lokacija u tekstu, konteksta u tekstu i značajki dokumenata iz Elasticsearch upita [4]. Drugim riječima, mordecai pokušava razumijeti tekst te na temelju svog razumijevanja odabrati najbolju lokaciju. Na posljetku, mordecai vraća samo jedan rezultat, onaj koji je on proglasio najboljim, za svaki prepoznati imenovani entitet.

3. Implementacija

Implementirani sustav temelji se na klijent-poslužitelj arhitekturi. Poslužiteljski dio implementiran je kao REST API (eng. Representational State Transfer Application Programming Interface) koristeći programski jezik Python, verzije 3.9.12, te radnog okvira Flask. Time je omogućena integracija s poslužiteljem jednostavnim API pozivima, bez potrebe za klijentskom stranom. Klijentski dio sustava razvijen je pomoću React biblioteke uz korištenje shadcn biblioteke komponenata (eng. component library). Klijentski dio je razvijen kako bi testiranje sustava bilo brže, ali i svima dostupno, pa iz tog razloga nije detaljnije razrađeno u ovom radu.



Slika 3.1. Detaljan prikaz slijeda izvođenja u metodama za geoparsiranje

Na slici 3.1. prikazani su svi koraci geoparsiranja te je za svaku metodu naznačeno koji dio sustava je zadužen za pojedini proces geoparsiranja. Na slici se može vidjeti kako su za spaCy biblioteku korištene verzije 3.6 i 3.8, dok je verzija Mordecai3 biblioteka 3.0.0a0. Za LLM je u ovom radu odabran model *gpt-4o-mini* OpenAI-a za sve metode, a komunikacija s modelom odvija se jednostavnim API pozivima. Na isti način odvija se i komunikacija s Elasticsearch indeksima koji se pretražuju u svim implementiranim metodama.

U narednim poglavljima opisuju se glavni dijelovi razvijenog koda, a kompletnom

projektu, sa uputama za lokalno pokretanje, može se pristupiti na github repozitoriju [5].

3.1. Elasticsearch

U ovom potpoglavlju opisuje se pokretanje Docker kontejera koristeći Elasticsearch sliku, kreiranje volumena za taj kontejner te kreiranje Elasticsearch indeksa koristeći podatke iz GeoNames baze i Registra geografskih imena.

```
1  set -e
2
3  geojson_dir="$PWD/geojson_index"
4  geonames_dir="$PWD/geonames_index"
5
6  mkdir -p "$geojson_dir" "$geonames_dir"
7
8  echo "Starting Docker container and data volumes for Elasticsearch..."
9  docker run -d -p 127.0.0.1:9200:9200 \
10   -e "discovery.type=single-node" \
11   -v "$geojson_dir":/usr/share/elasticsearch/data/geojson \
12   -v "$geonames_dir":/usr/share/elasticsearch/data/geonames \
13   elasticsearch:7.10.1
14
15 echo "Waiting for Elasticsearch to start..."
16 until curl -s http://localhost:9200 >/dev/null; do
17   sleep 2
18 done
19 echo "Elasticsearch is running."
```

Kod 3..1: Isječak koda iz skripte elasticsearch_setup.sh-pokretanje kontejnera

Na početku skripte, u prvoj liniji koda 3..1, dodana je naredba koja prekida daljnje izvođenje skripte u slučaju greške. Kako se dijelovi skripte oslanjaju na uspješno izvođenje prethodnih naredbi, ta naredba je vrlo korisna jer spriječava nepotrebno nakupljanje grešaka. Linije 3-6 kreiraju direktorije koji se koriste kao volumeni Docker kontejnera kako bi elasticsearch mogao trajno pohranjivati sve što mu je potrebno. Kako se ne bi izbrisale korisne stvari koje je elasticsearch spremio, u liniji 6 dodana je zastavica -p, što zapravo znači da će se direktoriji kreirati samo u slučaju da već ne postoje. Nakon toga je

potrebno pokrenuti Docker kontejner narebom u redcima 9-13. U liniji 9 navodi se *host* na kojem će Docker kontejner "živjeti". U ovom slučaju je to *localhost*, odnosno lokalno računalo. Također je navedeno da će kontejner slušati port 9200 lokalnog računala te ga prosljeđivati na isti taj port u kontejneru, što je važno jer Elasticsearch radi na portu 9200. U slučaju da je taj port zauzet na lokalnom računalu, jednostavnom promjenom ove naredbe se može koristiti bilo koji drugi slobodni port lokalnog računala. Nadalje, u liniji 10 postavljena je varijablu okruženja kako bi Elasticsearch znao da će se ova instanca pokrenuti kao samostalni čvor te neće tražiti druge čvorove u klasteru. Linije 11 i 12 pridružuju ranije kreirane direktorije Docker volumenima, a u liniji 13 se navodi koja će se slika Elasticsearch-a koristiti. Linije 16-18 su dodane kako se ne bi nastavljalo s izvršavanjem naredbi prije nego što kontejner bude pokrenut. To je iznimno važno jer bi u protivnom došlo do pogreške i izvršavanje skripte bi se zaustavilo, no kontejner bi ostao pokrenut. Dodatno, na isječku 3..1 može se primjetiti kako su dodane i *echo* naredbe koje korisnika izvještavaju o trenutnom stanju izvođenja.

```
1  # Update Elasticsearch disk settings
2  echo "Updating Elasticsearch disk settings..."
3  curl -X PUT "localhost:9200/_cluster/settings" -H 'Content-Type:
       application/json' -d'
4  {
5    "transient": {
6      "cluster.routing.allocation.disk.watermark.low": "10gb",
7      "cluster.routing.allocation.disk.watermark.high": "5gb",
8      "cluster.routing.allocation.disk.watermark.flood_stage": "4gb",
9      "cluster.info.update.interval": "1m"
10   }
11 }'
12
13 # --- Geonames Index Setup ---
14 echo "Downloading Geonames gazetteer..."
15 wget -q https://download.geonames.org/export/dump/HR.zip
16 wget -q https://download.geonames.org/export/dump/admin1CodesASCII.txt
17 wget -q https://download.geonames.org/export/dump/admin2Codes.txt
18
19 unzip -q HR.zip
20 echo "Geonames data unpacked."
21
```

```
22 echo "Creating mappings for the Geonames index..."
23 curl -XPUT 'localhost:9200/geonames' -H 'Content-Type: application/
       json' -d @geonames_mapping.json
24
25 echo "Loading Geonames data into Elasticsearch..."
26 python3 geonames_elasticsearch_loader.py
27
28 # Cleanup Geonames files
29 echo "Cleaning up Geonames downloaded files..."
30 rm -f HR.zip admin1CodesASCII.txt admin2Codes.txt HR.txt readme.txt
31
32 echo "Geonames setup complete."
```

Kod 3..2: Isječak koda iz skripte elasticsearch_setup.sh - kreiranje i populacija geonames indeksa

Isječak koda 3..2 preuzet je sa github repozitorija [6] prema opisu iz rada [4]. Jedine promjene u odnosu na preuzetu skriptu, u ovom isječku, su linije 15 i 19 jer su u ovom radu potrebne samo lokacije za Hrvatsku pa se ovim promjenama izbjeglo preuzimanje velike količine podataka koji neće biti korišteni. Isto tako je dodana i linija 30 koja, nakon korištenja, briše sve preuzete datoteke. Datoteka s podacima za mapiranje toponima (geonames_mapping.json) te skripta za populaciju indeksa (geonames_elasticsearch_loader.py) također su preuzete s istog github repozitorija.

U nastavku su objašnjeni dijelovi koji su samostalno razvijeni kako bi se u sustavu mogao koristiti i Registar geografskih toponima.

```
1  {
2    "mappings": {
3      "properties": {
4        "identifikator": { "type": "keyword" },
5        "vrstaobiljezja": { "type": "text" },
6        "geografskoime": {"type": "text"},
7        "jezik": { "type": "keyword" },
8        "status": { "type": "keyword" },
9        "geometry": {"type": "geo_point"}
10     }
11   }
```

```
}
```

Kod 3..3: Sadržaj geojson_mapping.json datoteke

U isječku 3..3 može se vidjeti mapiranje koje se koristi prilikom kreiranja indeksa za Registar geografskih imena. Geografsko ime i vrsta obilježja označena su kao tekst kako bi se po tim kriterijima moglo uspješnije pretraživati.

U kodu 3..4 može se vidjeti kako je u ovom radu implementirana populacija indeksa koristeći Registar geografskih imena, odnosno geojson datoteku preuzetu sa [7]. Također je važno napomenuti kako su koordinate u preuzetoj datoteci bile u HTRS96[1] referentnom koordinatnom sustavu, a Elasticsearch za koordinate prihvaća samo WGS84[2] referentni koordinatni sustav. Iz tog razloga napisana je skripta koja za svaki zapis u preuzetoj datoteci koordinate pretvara u WGS84 sustav. Ta skripta ovdje neće biti detaljnije razrađena jer će sličan kod biti opisan u nastavku.

```
with open("registar_geografskih_imena.geojson", "r", encoding="utf-8")
    as file:
    geojson_data = json.load(file)

features = geojson_data.get("features", [])
if not features:
    print("No features found in the GeoJSON file.")
    return

bulk_data = ""
for feature in features:
    action = {"index": {"_index": "geojson"}}
    document = {
        "identifikator": feature["properties"].get("identifikator"),
        "vrstaobiljezja": feature["properties"].get("vrstaobiljezja"),
        "geografskoime": feature["properties"].get("geografskoime"),
        "jezik": feature["properties"].get("jezik"),
        "status": feature["properties"].get("status"),
        "geometry": feature["geometry"]["coordinates"]
```

[1]HTRS96 je službeni položajni referentni koordinatni sustav Republike Hrvatske. U njemu su geografska širina i dužina iskazani u metrima.

[2]WGS84 je koordinatni sustav koji se najčešće koristi za prikazivanje geografske dužine i širine, a izražen je u stupnjevima.

```
19      }
20      bulk_data += f"{json.dumps(action)}\n{json.dumps(document)}\n"
21
22  headers = {"Content-Type": "application/x-ndjson"}
23  response = requests.post("http://localhost:9200/_bulk", headers=
        headers,data=bulk_data)
```

Kod 3..4: Isječak koda iz geojson_elasticsearch_loader.py - populacija indeksa

Opisani isječci koda koriste se u nastavku elasticsearch_setup.sh, kao što je vidljivo u 3..5

```
1  # --- GeoJSON Index Setup ---
2  echo "Creating mappings for the GeoJSON index..."
3  curl -XPUT 'localhost:9200/geojson' -H 'Content-Type: application/json
       ' -d @geojson_mapping.json
4
5  echo "Loading GeoJSON data into Elasticsearch..."
6  python3 geojson_elasticsearch_loader.py
7
8  echo "GeoJSON setup complete."
```

Kod 3..5: Isječak koda iz skripte elasticsearch_setup.sh - kreiranje i populacija geojson indeksa

U ovom isječku može se vidjeti kako se na zapravo vrlo jednostavan način kreira te populira indeks elasticsearcha. U liniji 3 nalazi se naredba koja pomoću API zahtjeva kreira indeks unutar Elasticsearcha, a zatim se pokreće skripta geojson_elasticsearch_loader.py koja na već opisani način obavlja populaciju.

Ovime su oba indeksa populirana te su spremna za pretraživanje, a primjeri dokumenata u oba indeksa mogu se vidjeti u isječcima 3..6 i 3..7

```
1  {
2      "_index": "geonames",
3      "_type": "_doc",
4      "_id": "3186886",
5      "_score": 7.4083824,
6      "_source": {
7          "geonameid": "3186886",
```

```
        "name": "Zagreb",
        "asciiname": "Zagreb",
        "alternativenames": [
            "Zagreb",
            "Zagrebas",
            "Zagrab",
            "Zagrebi",
            "Agram",
            "Zagabria"
        ],
        "coordinates": "45.81444,15.97798",
        "feature_class": "P",
        "feature_code": "PPLC",
        "country_code2": "HR",
        "country_code3": "HRV",
        "admin1_code": "21",
        "admin1_name": "Zagreb",
        "population": "663592",
        "modification_date": "2025-02-14"
    }
}
```

Kod 3..6: Primjer dokumenta u GeoNames bazi

```
{
    "_index": "geojson",
    "_type": "_doc",
    "_id": "NpGNAZUB4pg3cFD0XfwR",
    "_score": 7.630581,
    "_source": {
        "identifikator": "HRRGIIM00100718",
        "vrstaobiljezja": "hotel",
        "geografskoime": "Zagreb",
        "jezik": "Hrvatski",
        "status": "službeno",
        "geometry": [
        45.782838905197465,
        15.983985194817494
        ]
    }
```

```
}
```

Kod 3..7: Primjer dokumenta u Registru geografskih imena

3.2. Poslužitelj

U ovom potpoglavlju opisani su glavni dijelovi implementacije poslužitelja. Na početku se opisani dijelovi koda koji su koriste u više od jedne metode, a zatim su opisane implementacije samih metoda, kronološki kako su razvijane. Uz svaku metodu naznačene su njezine prednosti i mane.

```
@app.route('/process_excel', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return 'No file part', 400
    file = request.files['file']
    if file.filename == '':
        return 'No selected file', 400
    text = request.form.get('text')
    x = request.form.get('lat')
    y = request.form.get('long')
    parsing_method = request.form.get('method')
    database = request.form.get('database')
    tolerated_diff_value = request.form.get('toleratedDiff')
    tolerated_diff: float
    try:
        tolerated_diff = float(tolerated_diff_value)
    except ValueError:
        tolerated_diff = 1000.0
    normalize_entries = request.form.get('normalizeEntries', "false")
   == "true"
```

Kod 3..8: Isječak koda iz server.py - definiranje endpointa i parsiranje zahtjeva

Poslužitelj sadrži samo jedan *endpoint* koji prima zahtjeve POST metodom kako je prikazano u isječku koda 3..8 U isječku se mogu vidjeti i koji su sve očekivani parametri zahtjeva koji se šalje na taj *endpoint*. Odabir metode određuje se na temelju atributa *parsing_method*, *database* i *normalize_entries* poslanih u zahtjevu, a moguće vrijednosti za *parsing_method* i *database* prikazane su u isječcima 3..9 i 3..10

```
class ParsingType(Enum):
    MORDECAI = 'mordecai'
    OPENAI = 'openai'
    SPACY = 'spacy'
```

Kod 3..9: Isječak koda iz server.py - definiranje enuma za atribut *method* iz zahtjeva

```
class Database(Enum):
    GEOJSON = 'geojson'
    GEONAMES = 'geonames'
```

Kod 3..10: Isječak koda iz server.py - definiranje enuma za atribut *database* iz zahtjeva

U slučaju da je vrijednost *parsing_method* atributa *mordecai*, atributi *database* i *normalize_entries* se ignoriraju te se za geoparsiranje koristi mordecai biblioteka.

Nakon uspješnog parsiranja svih parametara zahtjeva koji je poslan, potrebno je obraditi excel datoteku te iz nje izvući potrebne informacije. Za to se koristi polars biblioteka, jedna od najpopularnijih biblioteka za obradu podataka.

```
def extract_text_and_coordinates(file: FileStorage, text_col: str,
    x_col: str, y_col: str) -> List[RowData]:
    file_content = io.BytesIO(file.read())
    df = pl.read_excel(file_content)

    required_columns = {text_col, x_col, y_col}
    if not required_columns.issubset(df.columns):
        missing_columns = required_columns - set(df.columns)
        raise ValueError(f"Missing columns in the Excel file: {
    missing_columns}")

    filtered_df = df.select([text_col, x_col, y_col])
    result = [RowData(text=row[text_col], location=Location(lat=row[
    x_col], lng=row[y_col])) for row in filtered_df.iter_rows(named=
    True)]

    return result
```

Kod 3..11: Isječak koda iz server.py - obrada excel datoteke

U isječku 3..11 se može vidjeti kako su u excel datoteci obavezna samo tri stupca. To

su stupac koji sadrži zapis koji je potrebno geoparsirati te stupci koji sadrže geografsku širinu i dužinu. U isječku 3..8 to su objekti *text*, *x* i *y* koji u zahtjevu dobivaju naziv[3] stupaca s očekivanim podatcima koji se zatim prosljeđuju metodi iz isječka 3..11 Ako bilo koji od tih parametara ne postoji, podiže se iznimka koja se zatim i vraća korisniku. Odnosno, korisnik dobije odgovor sa HTTP statusom 400 (Bad request) te tekstom koji mu objašnjava koji stupac nedostaje ili se ne može pronaći u datoteci. Tek ako su svi poslani parametri uspješno parsirani obrađuje se datoteka. To se radi na način da se zapis koji je potrebno geoparsirati sprema u atribut *text*, dok se geografska širina i dužina spremaju u atribut *location*. Na kraju metode vraća se lista takvih objekata koji se kasnije koriste za geoparsiranje te ocjenjivanje same uspješnosti geoparsiranja.

Nadalje, kako bi se ocijenila uspješnost geoparsiranja potrebno je izračunati udaljenost lokacija geoparsiranog toponima i lokacije koja je dana u zapisu. Za to se koristi funkcija prikazana u kodu 3..12

```
def calculate_difference(first: Location, second: Location) -> Union[
    Difference, None]:
    if second:
        difference_lat = abs(first.lat - second.lat)
        difference_lng = abs(first.lng - second.lng)
        return Difference(
            diff_lat=difference_lat,
            diff_lng=difference_lng,
            diff=math.hypot(difference_lat, difference_lng)
        )
    else:
        return None
```

Kod 3..12: Isječak koda iz server.py - računanje udaljenosti lokacija

U funkciji se prvo računaju udaljenosti po osima apscisa i ordinata, a zatim se prema pitagorinom poučku, koristeći ugrađeni modul *math*, računa ukupna udaljenost dviju točaka. Kako je za ocjenjivanje uspješnosti geoparsiranja dan prag u metrima, vrlo je važno da se i ova udaljenost izražava u metrima. U potpoglavlju 3.1. koordinate u zapisima mogu biti u dva referentna koordinatna sustava, pa je vrlo važno da se prije računanja udaljenosti uvjerimo da su koordinate zapisane u HTRS96 sustavu.

[3]Naziv stupca se u ovom kontekstu smatra prvi zapis u tom stupcu.

```
def transform_coordinates(coordinates: Location, source:
    ReferenceSystem = ReferenceSystem.WGS84, target: ReferenceSystem =
     ReferenceSystem.HTRS96) -> Location:
     transformer = Transformer.from_crs(source.value, target.value,
    always_xy=True)
     lng, lat = transformer.transform(coordinates.lng, coordinates.lat)
     return Location(lat=lat, lng=lng)
```

Kod 3..13: Isječak koda iz server.py - preračunavnje koordinata u drugi referentni koordinatni sustav

Za preračunavanje koordinata napisana je funkcija prikazana u kodu 3..13 Iako se trenutno u sustavu ova funkcija koristi samo za preračunavanje iz WGS84 u HTRS96 referentni koordinatni sustav, u potpis metode dodana su dva parametra kako bi se moglo proizvoljno odrediti izvorišni i željeni referentni sustav. To je napravljeno kako bi se mogli testirati i zapisi koji sadrže koordinate u nekom trećem referentnom koordinatnom sustavu.

Na sljedećem primjeru je prikazan isječak programskog koda koji se koristi za procesiranje geoparsiranih toponima.

```
def calculate_difference_and_rating_for_geolocations(geolocation_list:
     List[GeoLocation], entry: RowData, tolerated_diff: float):
     result_list = []
     reference_system = detect_reference_system(entry.location)
     for geolocation in geolocation_list:
         transformed_coordinates = transform_coordinates(Location(lat=
    geolocation.lat, lng=geolocation.lng))
         diff: Union[Difference, None]
         if reference_system == ReferenceSystem.HTRS96:
             diff = calculate_difference(entry.location,
    transformed_coordinates)
             geolocation.lat = transformed_coordinates.lat
             geolocation.lng = transformed_coordinates.lng
         else:
             transformed_provided_coordinates = transform_coordinates(
    entry.location)
             diff = calculate_difference(
    transformed_provided_coordinates, transformed_coordinates)
```

```
14          raiting: int
15          if diff is not None:
16              if diff.diff < tolerated_diff:
17                  raiting = 1
18              else:
19                  raiting = 0
20              result_list.append({
21                  "difference": diff.dict(),
22                  "geoparsed": geolocation.dict(),
23                  "rating": raiting
24              })
25      return result_list
```

Kod 3..14: Isječak koda iz server.py - Procesiranje geoparsiranog toponima

Metoda prikazana u isječku 3..14 koristi se za procesiranje i ocjenjivanje geoparsiranih toponima. Nakon što se zapis iz dobivene datoteke obradi, poziva se ova metoda s listom geoparsiranih toponima, dijelom zapisa iz datoteke (dobiveno koristeći kod 3..11) te toleriranom pogreškom u metrima (također dobiveno od korisnika). U liniji 3 se određuje u kojem referentnom koordinatnom sustavu su zapisane koordinate iz zapisa. Ovaj korak je vrlo važan jer označava je li te koordinate potrebno transformirati u drugi sustav. Ako referentni koordinatni sustav nije HTRS96, tada je potrebno odraditi transformiranje, ali samo za potrebe računanja udaljenosti. Naime, korisničko iskustvo bilo bi vrlo loše ako bi korisnik poslao koordinate u WGS84 referentnom koodinatnom sustavu, a u odgovoru dobio te iste koordinate u HTRS96 referentnom koordinatnom sustavu, s kojim možda niti nije upoznat. Kako se u bazama nalaze koordinate koje su sigurno u WGS84 referentnom koordinatnom sustavu, u liniji 5 obavlja se transformiranje koordinata za svaki geokodirani toponim kako bi se za njega mogla izračunati udaljenost. Nakon toga se poziva metoda za računanje udaljenosti te se u linijama 16-19 određuje uspješnost samog geoparsiranja toponima. Važno je za napomenuti i kako se, u slučaju da zapis sadrži koordinate u HTRS96 referentnom koordinatnom sustavu, koordinate geoparsiranog toponima također transformiraju u HTRS96 kako bi sve informacije u odgovoru bile dosljedne. Na posljetku se kreira objekt koji sadrži sve informacije o geoparsiranom toponimu (u atributu *geoparsed* pohranjene su informacije o geoparsiranom toponimu dobivene jednom od kasnije objašnjenih metoda, atribut *difference* sadrži udaljenosti po

osima kao i ukupnu udaljenost, a atribut *rating* sadrži informaciju o uspješnosti geoparsiranja) te se on dodaje u listu koja se nakon obrađivanja svih geoparsiranih toponima vraća iz funkcije.

3.2.1. Mordecai

U ovom poglavlju opisuje se implementacija geoparsiranja koristeći mordecai biblioteku te su objašnjeni nedostatci ove biblioteke.

```
geo = Geoparser(debug=True)
```

Kod 3..15: Isječak koda iz server.py - Inicijalizacija mordecai geoparsera

Za početak je potrebno napraviti inicijalizaciju, odnosno stvaranje objekta, mordecaievog geoparsera što je prikazano u isječku 3..15 Ovo je definirano na razini cijele aplikacije, odnosno radi se samo jednom, pri pokretanju poslužitelja. Kako mordecai interno koristi spaCy biblioteku za prepoznavanje imenovanih entiteta, najbolja opcija bila bi korištenje hrvatskog modela za spaCy. To, nažalost, u trenutku pisanja ovog rada nije moguće. Naime, prilikom korištenja hrvatskog modela, mordecai biblioteka interno generira iznimku, a to rezultira iznimkom u samom poslužitelju. To se događa jer niti jedan hrvatski model nema sve komponente koje su mordecai biblioteci potrebne (glavni problem je nedostatak *transformer* komponente). Moguće je razviti vlastiti transformer te ga dodati u cjevovod moredecaia, ali to nije rađeno u sklopu ovog rada.

```
translated_excel_data = translate_entries_to_english(
    formated_excel_data)
     for index in range(len(translated_excel_data)):
         entry = translated_excel_data[index]
         geolocation_list = geoparse_text_and_map_result(entry.text)
         entry_data_list =
    calculate_difference_and_rating_for_geolocations(geolocation_list,
     entry, tolerated_diff)

         data.append({
             "requested": formated_excel_data[index].dict(),
             "geoparsed_results": entry_data_list
         })
```

Kod 3..16: Isječak koda iz server.py - Korištenje mordecai biblioteke za geoparsiranje

U isječku 3..16 može se vidjeti implementirano geoparsiranje pomoću mordecai biblioteke. Kako nije moguće koristiti hrvatski model za prepoznavanje imenovanih entiteta, potrebno je sve zapise prevesti na engleski, što je i odrađeno u prvoj liniji isječka. Objekt *formated_excel_data* dobiven je pozivanjem metode u isječku 3..11, a za prevođenje je korišten OpenAI API koji se pokazao vrlo točan u izvršavanju ovog zadatka. Za ovaj, ali i sve ostale zadatke za koje je korišten OpenAI API, korišten je model "gpt-4o-mini". Nako prevođenja potrebno je za svaki objekt u listi odraditi geoparsiranje te izračunati udaljenost geoparsiranog toponima i koordinata koje su dobivene u excel datoteci za što se koristi funkcija iz isječka 3..12 Iako mordecai vraća samo jedan objekt prilikom geoparsiranja, sustav taj objekt dodaje u listu kako bi se izjednačilo ponašanje s ostalim metodama koje se koriste. Na posljetku se ti rezultati dodaju u listu objekata pod nazivom *data* koji predstavlja odgovor sustava.

Glavni problem korištenja ove metode već je objašnjen, a to je nemogućnost korištenja hrvatskog modela za prepoznavanje imenovanih entiteta, ali nažalost to nije jedini problem. Isto tako nije moguće koristiti niti najnoviju verziju spaCy biblioteke. Naime zadnja verzija spaCy biblioteke koja se može koristiti je 3.6.1, dok je najnovija verzija 3.8.0 u trenutku pisanja ovog rada. Razlog za to je promjena odgovora koji vraća metoda prepoznavanja imenovanih entiteta unutar spaCy biblioteka. Iako nevjerojatno, takva promjena napravila se u *minor* verziji biblioteke. Razvojni tim mordecai biblioteke prepoznao je ovaj problem, ali u trenutku razvoja sustava još nije izdana s rješenjemm ovog problema. To uzrokuje lošije rezultate koji će detaljnije biti prikazani u poglavlju 4.3. Zbog ovih problema dodana je metoda koja koristi spaCy biblioteku bez mordecaia što će biti detaljnije razrađenu u sljedećem potpoglavlju.

3.2.2. SpaCy

Kako je korištenje mordecai biblioteke dalo vrlo loše rezultate prilikom testiranja, razvijeno je rješenje koje koristi spaCy biblioteku za prepoznavanje imenovanih entiteta, a zatim pretražuje elasticsearch indeks kako bi se odradilo geoparsiranje. Za to se koristila zadnja, trenutno dostupna, verzija spaCy biblioteke (3.8.0) uz korištenje hrvatskog modela kako je i prikazano u isječku 3..17

```
nlp = spacy.load('hr_core_news_lg')
```

Kod 3..17: Isječak koda iz server.py - Inicijalizacija jezičnog modela korištenjem spaCy biblioteke

Inicijalizacija je, kao i u za mordecai biblioteku, definirana na razini cijele aplikacije. Kreirani jezični model se koristi za prepoznavanje imenovanih entiteta kao što je prikazano u isječku 3..18

```
def geoparse_text(text: str) -> List[str]:
    spacy_doc = nlp(text)

    geoparsed_entry_list = [ent.text for ent in spacy_doc.ents if ent.label_ in {"GPE", "LOC"}]
    return geoparsed_entry_list if len(geoparsed_entry_list) > 0 else [text]
```

Kod 3..18: Isječak koda iz server.py - Prepoznavanje imenovanih entiteta

U drugoj liniji odrađeno je prepoznavanje imenovanih entiteta u tekstu koji je proslijeđen ovoj metodi. Nakon toga u liniji četiri napravljeno je filtriranje imenovanih entiteta na način da se zadrže samo imena lokacija (u kodu LOC) i geopolotičkih entiteta (u kodu GPE). Geopolitički entiteti su države, gradovi, regije i slično dok u lokacije spadaju sve geografske lokacije koje ne spadaju u geopolitičke entitete, na primjer rijeke, planine i slično. SpaCy biblioteka dala je loš postotak prepoznatih imenovanih entiteta u slučajevima kada se obrađuju samo toponimi, odnosno kada tekst koji se analizira sadrži samo jedan toponim (npr. "Zagreb"). Zato je u liniji 5 napravljena je prilagodba koja, u slučaju da nijedan imenovani entitet nije prepoznat, vraća cijeli tekst. Takva dorada uvelike poboljšava performanse sustava jer ne rezultira lažno pozitivnim rezultatima. Odnosno, ako je kao tekst korišten sami toponim, on će se vratiti kao rezultat ove metode te će se on koristiti za pretraživanje indeksa, a u slučaju da je korišten kompleksniji zapis, ali nije prepoznat niti jedan entitet, pretraživanje indeksa neće dati rezultate.

Nakon određivanja svih imenovanih lokacija i geopolitičkih entiteta, za potrebne geokodiranje, pretražuje se indeks elasticsearcha.

```
query = {
    "query": {
```

```
3           "match": {
4               "geografskoime": search_string,
5               "fuzziness": 1
6           }
7       },
8       "size": 20
9   }
```

Kod 3..19: Isječak koda iz server.py - Upit za pretraživanje indeksa

U isječku 3..19 prikazan je upit kojim se pretražuje indeks elasticsearcha u kojem su podaci iz Registra geografskih imena, ali na analogan način pretražuje se i indeks koji sadrži podatke iz GeoNames baze. Kao što se može vidjeti, uzima se prvih 20 rezultata, a pretraga se radi samo po imenu u dokumentu, odnosno po atributu *geografskoime*. Pretraga u elasticsearchu vraća dokumente rangirane prema sličnosti s traženim pojmom, a vraćaju se samo dokumenti koji zadovoljavaju kriterij pretrage. Elasticsearch prije pretrage odrađuje analizu upita te razbija traženi pojam na riječi, a tek nakon toga pretražuje dokumente po riječima razbijenog atributa. Drugim riječima, ako se elasticsearch pretražuje toponimom "Biograd na Moru", elasticsearch će to razdvojiti na ["Biograd", "na", "Moru"] te vratiti rangirane dokumente koji sadrže barem jednu od tih riječi. Naravno najviše rangiran će biti dokument čije je ime "Biograd na Moru". Ovaj princip je vrlo važan jer spaCy biblioteka prilikom prepoznavanja imenovanih entiteta u toponimu "Biograd na Moru" kao imenovani entitet prepoznaje samo "Biograd", ali zahvaljujući naprednoj pretrazi elasticsearcha i to je dovoljno za uspješnu pretragu koristeći ovaj jednostavan upit. Dodatno, u atribut je dodana i linija 5 koja tolerira grešku u jednom slovu, odnosno za tražene atribute "Zagrebu", "Zagrb" i "Zagrebb" pretraga će vratiti dokument s imenom "Zagreb".

Do problema dolazi kada se koriste toponimi koji nisu u nominativu, što je često u svakodnevnom govoru hrvatskog jezika. Naime, u slučaju kada treba geoparsirati zapis "Šetao sam Zagrebom." spaCy će prilikom prepoznavanja imenovanih entiteta prepoznati "Zagrebom", ali pretraga elasticsearcha neće dati rezultate. Naravno i taj dio se može riješiti u samom upitu tako da se poveća *fuzziness*, ali to bi moglo rezultirati i vraćanjem pogrešnih rezultata. Iz tog razloga dodan je još jedan korak između prepoznavanja imenovanih entiteta i pretrage indeksa.

3.2.3. SpaCy uz normalizaciju

Zbog problema koji su opisani u prethodnom potpoglavlju, implementirana je normalizacija svih prepoznatih imenovanih entiteta. Za to se koristi OpenAI API kojem se šalje lista svih imenovanih entieta za neki zapis, a njegov zadatak je vratiti nominativ tih entiteta. Ovaj pristup pokazao se vrlo uspješan, kako je vidljivo u poglavlju 4. Jedini problem predstavljala je kreativnost korištenog modela. Tako je u nekim slučajevima model umjesto vraćanja nominativa nekog toponima potpuno promijenio toponim. Na primjer za toponim "Savišće" model je vratio "Savičica" što niti nije postojeća lokacija u Hrvatskoj. Instrukcije za API bile su vrlo detaljne, ali korišteni model nije dovoljno napredan da bi za svaki toponim vratio ispravne rezultate.

Nakon normalizacije koristeći OpenAI, ponovno se pretražuje indeks upitom prikazanim u kodu 3..19

3.2.4. OpenAI

Zbog uspješnosti normalizacije zanimljivo je bilo istraživanje mogućnosti koje pruža OpenAI API. Iz tog razloga napravljena je i metoda koja ne koristi spaCy biblioteku za prepoznavanje imenovanih entiteta, već toponime prosljeđuje OpenAI API-u koji zatim sam radi prepoznavanje imenovanih entiteta te normalizaciju istih. Ova metoda pokazala se iznenađujuće uspješnom, imajući na umu da model nije treniran specifično za takve zadatke. Ipak ponovno se pojavljuju problemi koji su već opisani u prethodnom potpoglavlju.

4. Rezultati testiranja

Aplikacija je testirana pomoću četiri testna skupa podataka koji potječu iz raznih bioloških istraživanja. Svaki skup podataka sadrži opis lokacije, geografsku dužinu (oznaka x) i geografsku širinu (oznaka y). Važno je napomenuti kako su za geografske koordinate korištena dva različita referentna koordinatna sustava WGS84[1] i HTRS96[2], ovisno o skupu. Prvi skup podataka (u daljnjem tekstu SKUP1) sadržava podatke o 246 lokacija u Zagrebačkoj županiji na kojima su zabilježene invazivne biljne vrste [8] . Lokacije u ovom skupu podataka čine informacije o ulicama, nerijetko i o kućnom broju te mjestu uzorkovanja. Drugi skup podataka (u daljnjem tekstu SKUP2) sadržava podatke o lokacijama 224 primke graha unutar Hrvatske baze podataka o biljnim genetskim izvorima [9]. Podaci o lokacijama su jednostavni i kratki bez detaljnih opisa složene strukture. Treći skup podataka (u daljnjem tekstu SKUP3) sadrži lokacijske podatke 389 dojava građana o nalazima krijesnica u Hrvatskoj putem građanske inicijative Krešo Krijesnica [10]. Lokacije u ovom skupu podataka su ukratko opisane korištenjem jednog toponima u nominativu. Četvrti skup podataka (u daljnjem tekstu SKUP4) potječe iz istraživanja o rasprostranjenosti različitih vrsta rakova u Hrvatskoj te sadrži 341 lokaciju kompleksne strukture [11]. Primjer zapisa u svakom od opisanih skupova dan je u tablici 4.1.

[1] WGS84 je koordinatni sustav koji se najčešće koristi za prikazivanje geografske dužine i širine, a izražen je u stupnjevima.

[2] HTRS96 je službeni položajni referentni koordinatni sustav Republike Hrvatske. U njemu su geografska širina i dužina iskazani u metrima.

Skup podataka	Opis lokacije	X	Y	Referentni koordinatni sustav
SKUP1	Braće Radić 13, Dugo Selo	478013	5074126	HTRS96
SKUP2	Tovarnik	45.16	19.15	WGS84
SKUP3	Gornje Vrapče	15.8967573	45.8185691	WGS84
SKUP4	PP Papuk, Jankovacki potok , 100 m nizvodno od slapa	593850.9	5044337	HTRS96

Tablica 4.1. Primjer zapisa u skupovima podataka

Dodatno, za potrebe geoparsiranja testirane su dvije baze geografskih imena. Baza GeoNames [12] i Registar geografskih imena, službeni popis toponima za područje RH koji broji preko 120 000 službenih hrvatskih geografskih imena [7]

Razlika (udaljenost) danih i geoparsiranih koordinata lokacija korištena je kao mjerilo uspješnosti geoparsiranja te je za potrebe ovog testiranja postavljen strogi prag od 1000 m. Drugim riječima, ako je udaljenost manja od 1000 metara geoparsiranje je označeno kao uspješno, u protivnom je označeno kao neuspješno. Općenito je za različite slučajeve potrebna različita razina preciznosti, zbog toga je prag dodan kao jedan od ulaznih parametara te ga korisnik proizvoljno postavlja.

4.1. SpaCy

U ovom poglavlju opisuju se rezultati testiranja dviju metoda korištenjem spaCy biblioteke. Točnije, opisuju se rezultati testiranja prilikom korištenja spaCy biblioteke te rezultati testiranja prilikom korištenja spaCy bibilioteke uz dodatnu normalizaciju pomoću OpenAI API-a. Metode su, usporedbe radi, testirane na obje baze navedene u prethodnom poglavlju.

4.1.1. SpaCy uz Registar geografskih imena

U sljedećoj tablici prikazane su glavne karakteristike rezultata testiranja spaCy biblioteke uz Registar geografskih imena.

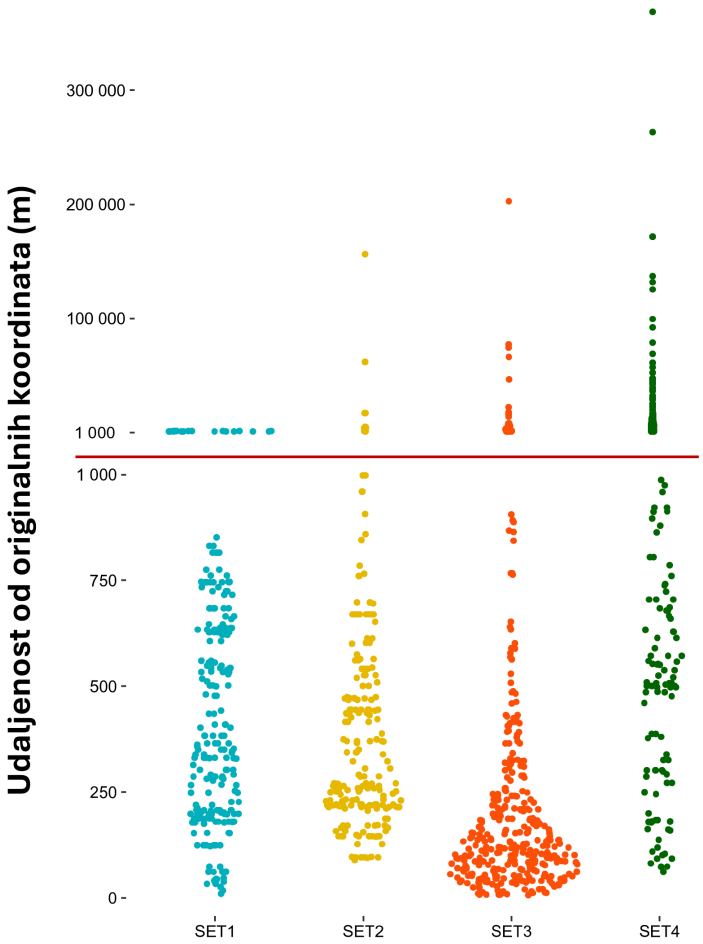
Korišteni skup podataka	SKUP1	SKUP2	SKUP3	SKUP4
Ukupno zapisa	246	224	389	341
Postotak geoparsiranih zapisa	100%	100%	99%	94%
Postotak uspješno geoparsiranih zapisa	91%	89%	87%	30%
Minimalna udaljenost (m)	10.4	90	7	62
Maksimalna udaljenost (m)	1,403	156,372	203,004	369,012
Prosječna udaljenost (m)	472	2,820	1,889	14,033

Tablica 4.2. Rezultati geoparsiranja

Iz tablice 4.2., prema postotku geoparsiranih i uspješno geoparsiranih zapisa, jasno se može vidjeti kako je sustav bio najuspješniji prilikom geoparsiranja SKUP1 podataka. Takav rezultat je i očekivan, uzevši u obzir kako ovaj skup podataka ima najdetaljnije opise lokacije. Iako je za SKUP2 i SKUP3 prosječna udaljenost mnogo veća nego za SKUP1, to ne znači da je sustav lošiji u geoparsiranju tih podataka. Naime, velika maksimalna udaljenost u ovim skupovima uzrokovana je podacima u bazi podataka. Kako u Hrvatskoj postoje istoimeni toponimi u raznim dijelovima zemlje, a baza podataka ne sadrži sve toponime, moguće je da sustav pronađe "pogrešan" toponim. Jedan takav primjer je toponim Marinci koji postoje u Hrvatskom Zagorju i u Istri. U SKUP2 sadržan je toponim Marinci u Istri, ali u Registru geografskih imena nalazi se samo istoimeni toponim u Hrvatskom Zagorju. Iz tog razloga je sustav za Marince u Istri vratio pogrešnu lokaciju uz veliku pogrešku od 156,372 metra. Takvih primjera ima ukupno šest u tom skupu podataka. Kad bi se ti zapisi uklonili iz skupa, prosječna udaljenost smanjila bi se na 755 metara, što je smanjenje od čak 73%. U SKUP3 se pojavljuje druga situacija koja utječe na točnost rezultata. Riječ je o točnosti hrvatskog modela za spaCy biblioteku. Na primjer, u toponimu "Kašinska ulica, Zagreb" biblioteka prepoznaje samo "Zagreb", tako da se detaljnija informacija lokacije izgubila, što rezultira neuspješnim geoparsiranjem. SKUP4 pokazao se kao najlošiji od četiri testirana skupa podataka. To se pripisuje karak-

teristikama zapisa koje sadrži. Naime, zapisi u SKUP4 nemaju dijakritike, sadrže kratice (ul. umjesto ulica), nerijetko imaju i razmake na krivim mjestima ("uzvodno do mjesta Ce ljakovci"), a uz sve to sadrže padeže geografskih imena ("1 km uzvodno od Slatinskog Drenovca"). Iz tih razloga ovaj skup daje rezultate sa velikom pogreškom, a to rezultira najmanjim ukupnim postotkom uspješno geoparsiranih zapisa.

Potaknut primjerom iz SKUP2, u kojem mali broj neuspješno geoparsiranih zapisa rezultira drastičnim povećanjem prosječne vrijednosti udaljenosti, te kako su za prva tri skupa postotci uspješno geoparsiranih zapisa relativno bliski, na slici 4.6. prikazane su udaljenosti za sve geoparsirane zapise uz vizualnu granicu između uspješno i neuspješno geoparsiranih zapisa. Prezicnije, crvenom horizontalnom linijom prikazan je prag koji dijeli zapise na uspješno geoparsirane i na neuspješno geoparsirane.



Slika 4.1. Prikaz udaljenosti od originalnih kooordinata za sve geoparsirane zapise

Lako je uočljivo kako su udaljenosti za uspješno geoparsirane zapise najviše grupi-

rane oko nule za SKUP3. Drugim riječima, gledajući samo uspješno geoparsirane zapise, tj. zapise ispod praga, za SKUP3 dobivaju se najtočniji rezultati. Zanimljivo je istaknuti kako je prosječna udaljenost uspješno geoparsiranih zapisa za SKUP3 (178 metara) manja za čak 225.8 metara od prosječne udaljenosti uspješno geoparsiranih zapisa za SKUP1 (403.9 metara), nominalno najuspješnijeg skupa za ovu metodu. Ipak, važno je naglasiti da SKUP3 sadrži više zapisa od SKUP1 pa time i veći broj uspješno geoparsiranih zapisa te iz tog razloga povremene veće udaljenosti imaju manji utjecaj na prosječnu vrijednost. Među neuspješno geoparsiranim zapisima ističu se zapisi iz SKUP1. Jasno je vidljivo kako su jedino neuspješno geoparsirani zapisi za SKUP1 izrazito grupirani oko praga. Odnosno, manjim povećanjem praga, SKUP1 bi rezultirao najvećim povećanjem uspješnosti.

4.1.2. SpaCy uz Registar geografskih imena i normalizaciju

Kao što je već ranije navedeno, zbog zapisa koji ne sadrže toponime u normaliziranom obliku, odnosno sadrže lokacije koje nisu u nominativu, uveden je dodatan korak normalizacije korištenjem OpenAI API-a. Karakteristike rezultata testiranja uz dodatno korištenje OpenAI API-a za normalizaciju prikazane su u sljedećoj tablici.

Korišteni skup podataka	SKUP1	SKUP2	SKUP3	SKUP4
Ukupno zapisa	246	224	389	341
Postotak geoparsiranih zapisa	100%	100%	97%	97%
Postotak uspješno geoparsiranih zapisa	91%	88%	85%	40%
Minimalna udaljenost (m)	10.4	90	7	24
Maksimalna udaljenost (m)	1,403	156,372	203,004	152,161
Prosječna udaljenost (m)	472	3,435	2,425	5,754

Tablica 4.3. Rezultati geoparsiranja

Iz tablice 4.3. možemo vidjeti kako su za SKUP4, koji ima najviše nenormaliziranih zapisa, poboljšani i postotak geoparsiranih zapisa (za 3.4%), ali i postotak uspješno geoparsiranih zapisa (za čak 36.6%). S obzirom da ne možemo uvijek očekivati jedino kvalitetne zapise, sa dijakritikama i razmacima na samo ispravnim mjestima, ovo je vrlo

dobar rezultat. Analizom rezultata ostalih skupova podataka može se vidjeti kako je broj, a samim time i postotak, uspješno geoparsiranih zapisa za SKUP1 ostao nepromijenjen, a za SKUP2 je smanjen (za 1%) kao i za SKUP3 (za 1.8%). To je neželjeno ponašanje uzrokovano korištenjem OpenAI API-a. Naime, za neke zapise model OpenAI-a promjenio je samu lokaciju dodavanjem ili izostavljanjem nekih slova, a u nekim slučajevima čak je i kompletno promijenio riječ. Tako je na primjer za toponim "Ferenšćica" u nekim slučajevima vratio "Ferenica", a toponim "Dotrošćina" kompletno je promijenio u "Dotrošinčina". Važno je napomenuti kako su oba ova toponima u nominativu i tu nije bilo nikakve potrebe za izmjenama istih.

4.1.3. SpaCy uz GeoNames

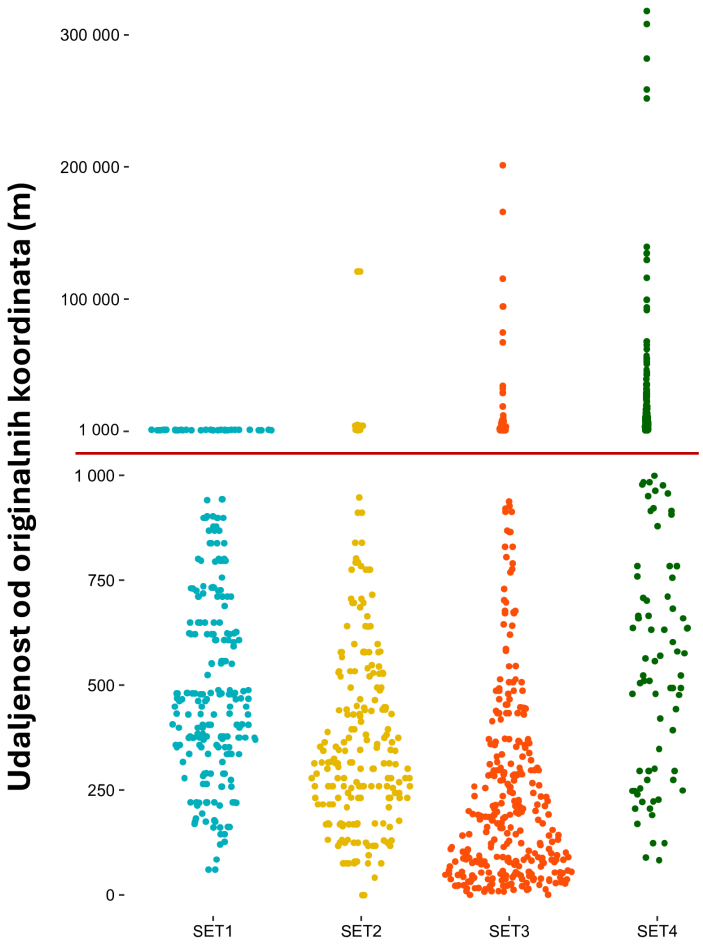
U ovom potpoglavlju analizirat će se uspješnost iste metode kao i u potpoglavlju 4.1.1., ali koristeći drugi izvor podataka za pretraživanje prepoznatih lokacija u zapisima. Karakteristike rezultata korištenjem spaCy bibilioteke uz GeoNames bazu prikazane su u sljedećoj tablici.

Korišteni skup podataka	SKUP1	SKUP2	SKUP3	SKUP4
Ukupno zapisa	246	224	389	341
Postotak geoparsiranih zapisa	91%	98%	95%	91%
Postotak uspješno geoparsiranih zapisa	87%	91%	80%	23%
Minimalna udaljenost (m)	60	0	312	83
Maksimalna udaljenost (m)	1,584	121,251	201,442	318,106
Prosječna udaljenost (m)	598	2,172	3,270	17,912

Tablica 4.4. Rezultati geoparsiranja

Kako se koristi ista metoda za prepoznavanje imenovanih entiteta kao i u potpoglavlju 4.1.3., indeks se pretraživao istim upitima. Postotak geoparsiranih zapisa je u odnosu na Registar geografskih imena manji za sve testne skupove podataka. U SKUP1 postotak geoparsiranih zapisa niži je za čak 8.5%, odnosno 21 zapis manje je geoparsiran. Postotak uspješno geoparsiranih zapisa također je manji za SKUP1 (za 4%), SKUP3 (za 7%) i SKUP4 (za 7%), ali za SKUP2 to nije slučaj. U SKUP2 je došlo do povećanja od

2%, odnosno četiri zapisa više su uspješno geoparsirana, a čak je pronađen i rezultat koji geolokacijom točno odgovara traženom zapisu. To znači da su za taj skup podataka, podaci koji se nalaze u GeoNames bazi u prosjeku točniji od onih koji se nalaze u Registru geografskih imena. Dodatno, baza GeoNames sadrži šest spornih zapisa za SKUP2 koji nedostaju u Registru geografskih imena, a to rezultira smanjenjem maksimalne pa i prosječne udaljenosti za taj skup.



Slika 4.2. Prikaz udaljenosti od originalnih kooordinata za sve geoparsirane zapise

Na slici 4.2. se može vidjeti kako su, kao i u potpoglavlju 4.1.1., udaljenosti za uspješno geoparsirane zapise najviše grupirane oko nule za SKUP3, no rezultati su ipak malo lošiji nego prilikom korištenja Registra geografskih imena. Samim pogledom na sliku 4.2., za uspješno geoparsirane zapise je vidljivo veće raspršenje udaljenosti po osi ordinata. Zbog toga su se povećale i prosječne udaljenosti uspješno geoparsiranih zapisa za same skupove. Prosječna udaljenost uspješno geoparsiranih zapisa za SKUP1

narasla je na 490.9 metra, odnosno povećala se za čak 21.5%, dok je prosječna udaljenost za SKUP3 porasla na 234.3 metra, što je povećanje od 31.6%. Posljedično se i razlika između prosječnih udaljenosti za uspješno geoparsirane zapise iz SKUP1 i SKUP3 povećavala na 256.6 metara, što je povećanje od 13.5% (30.8 metara). Odnosno, SKUP1 je izraženije lošiji za uspješno geoparsirane zapise u odnosu na SKUP3 nego što je bio pri korištenju Ragistra geografskim imena. Trend povećanja prosječne udaljenosti uspješno geoparsiranih zapisa vidljiv je i na SKUP4, gdje je prosječna udaljenost uspješno geoparsiranih zapisa porasla za 13.4%. Jedino se za SKUP2 može uočiti veće grupiranje oko manjih udaljenosti, a to potvrđuje i prosječna udaljenost koje je manja za 2.3%. Za neuspješno geoparsirane zapise, osim zapisa iz SKUP1, koji su ponovno grupirani oko praga od 1000 metara, može se uočiti pomalo drukčiji uzorak za zapise iz SKUP2. Sada se za neuspješno geoparsirane zapise iz SKUP2 pojavljuje blago grupiranje oko praga, ali također i veći broj zapisa za koje je dobivena udaljenost od originalnih kooordinata veća od 100 000 metara.

4.1.4. SpaCy uz GeoNames i normalizaciju

U ovom potpoglavlju, kao i u prethodnom, analizira se već korištena metoda (potpoglavlje 4.1.2.), ali na GeoNames bazi. Karakteristike rezultata testiranja prikazane su u sljedećoj tablici.

Korišteni skup podataka	SKUP1	SKUP2	SKUP3	SKUP4
Ukupno zapisa	246	224	389	341
Postotak geoparsiranih zapisa	91%	98%	93%	78%
Postotak uspješno geoparsiranih zapisa	87%	90%	78%	24%
Minimalna udaljenost (m)	60	0	305	83
Maksimalna udaljenost (m)	1,584	121,251	201,442	153,065
Prosječna udaljenost (m)	598	2,333	3,421	10,322

Tablica 4.5. Rezultati geoparsiranja

Iz tablice 4.5. može se vidjeti kako je uvođenje normalizacije rezultiralo podjednakim ponašanjem kao i u 4.1.2. Točnije, zbog već objašnjenog ponašanja modela OpenAI-

a, postotak geoparsiranih podataka, uz korištenje normalizacije smanjio se za SKUP1, SKUP2 i SKUP3, ali ovog puta smanjio se i za SKUP4 (za 14.1%). Za uspješno geoparsirane podatke ponovilo se ponašanje iz 4.5. Odnosno, postotak uspješno geoparsiranih zapisa za SKUP1 nije se promijenio, za SKUP2 smanjio se za 12.4%, a za SKUP3 se smanjio za 2.2%. Za SKUP4 postotak uspješno geoparsiranih zapisa ponovno se povećao, ali ovog puta za samo 6.5%. Uzevši opisane razlike u obzir, može se konstatirati kako je normalizacija postigla veći učinak prilikom korištenja Registra geografskih imena.

4.2. OpenAI

U ovom poglavlju opisuju se rezultati testiranja nove metode. Umjesto korištenja spaCy biblioteke za prepoznavanje imenovanih entiteta, korišten je model OpenAI-a. Kao i u prethodnom poglavlju, testiranje će biti provedeno na oba indeksa.

4.2.1. OpenAI uz Registar geografskih imena

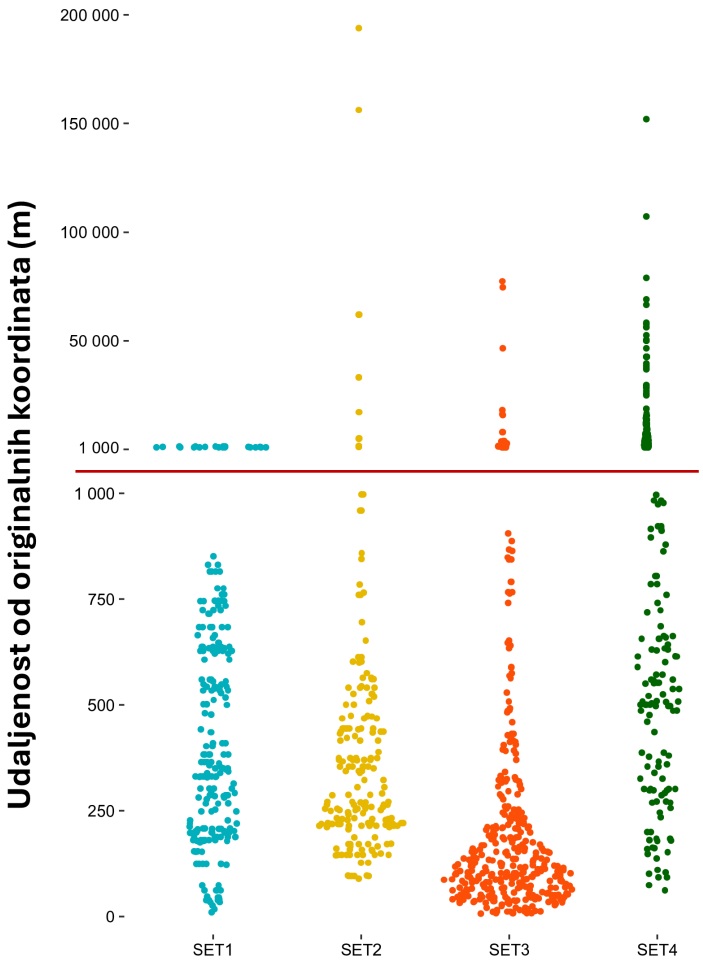
U sljedećoj tablici prikazane su glavne karakteristike rezultata testiranja modela OpenAI-a za prepoznavanje imenovanih entiteta uz Registar geografskih imena.

Korišteni skup podataka	SKUP1	SKUP2	SKUP3	SKUP4
Ukupno zapisa	246	224	389	341
Postotak geoparsiranih zapisa	100%	92%	97%	99%
Postotak uspješno geoparsiranih zapisa	91%	83%	89%	35%
Minimalna udaljenost (m)	10	90	7	62
Maksimalna udaljenost (m)	1,403	194,059	77,525	152,161
Prosječna udaljenost (m)	472	4,318	1,004	8,378

Tablica 4.6. Rezultati geoparsiranja

Iako model koji je korišten nije treniran specifično za prepoznavanje imenovanih entiteta, iz tablice 4.6. može se vidjeti kako su rezultati iznenađujuće dobri. Dio neuspješnih geoparsiranja dakako je uzrokovan već opisanim pogreškama u zapisima, ali veći dio je ipak uzrokovan kreativnošću OpenAI modela. Točnije, kako model nije treniran

za prepoznavanje imenovanih entiteta, iako mu je u danim uputama zabranjeno da promijeni lokaciju ili vraća onu koja ne postoji, on je ponekad vraćao imena nepostojećih lokacija. Na primjer, za toponim "Savišće" API bi vratio "Savičica". Prema rezultatima prikazanim u tablici 4.6. SKUP1 pokazao se kao najuspješniji korišteni skup podataka, kao i u potpoglavlju 4.1.1. gdje se opisivalo testiranje spaCy biblioteke uz pretraživanje istog indeksa. Po postotku geoparsiranih zapisa na drugom mjestu ovog puta se nalazi SKUP4 sa 99% geoparsiranih zapisa, a slijede ga SKUP3 sa 97% te SKUP2 sa 92%. Zanimljivo je kako se SKUP4 uz OpenAI pozicionirao na drugo mjesto prema postotku geoparsiranih podataka, dok je uz korištenje spaCy biblioteke te pretraživanjem istog indeksa bio na posljednjem mjestu. Iz toga se može zaključiti da je model uspio prepoznati imenovane entitete čak i uz neispravne dijakritičke znakove i pogrešno pozicionirane razmake. Analiziranje uspješno geoparsiranih zapisa također daje zanimljive rezultate. U nastavku će se rezultati uspoređivati sa rezultatima iz potpoglavlja 4.1.1. SKUP1 ponovno je postigao najveći postotak geoparsiranih zapisa (91%, identično rezultatima za spaCy biblioteku). Drugi po redu je SKUP3 s ostvarenih 89% uspješno geoparsiranih podataka što je povećanje od 2.4%. Povećanje u postotku uspješnosti postigao se i za SKUP4 u iznosu od 16.8%, ali to je ponovno najlošiji rezultat. Jedino smanjenje u postotku uspješno geoparsiranih zapisa pojavljuje se za SKUP2 u iznosu od 7.5%. Uzrok toga nije pronalaženje pogrešne lokacije, već ne prepoznavanje imenovanih entiteta za toponime poput "Lučelnice" i "Grešćevina".



Slika 4.3. Prikaz udaljenosti od originalnih kooordinata za sve geoparsirane zapise

Prikaz udaljenosti za uspješno geoparsirane zapise korištenjem modela OpenAI-a skoro identično odgovara prikazu udaljenosti za uspješno geoparsirane zapise korištenjem spaCy biblioteke sa slike 4.1. Udaljenosti uspješno geoparsiranih zapisa za SKUP3 ponovno su najviše grupirane oko nule te je prosječna udaljenost uspješno geoparsiranih zapisa za SKUP3 ponovno manja od prosječne udaljenosti uspješno geoparsiranih zapisa za SKUP1, nominalno najuspješnijeg skupa. Neuspješno geoparsirani zapisi za SKUP1 i SKUP2 također skoro identično odgovaraju neuspješno geoparsiranim zapisima sa slike 4.1. Odnosno, neuspješno geoparsirani zapisi za SKUP1 su grupirani oko praga, a za neuspješno geoparsirane zapise iz SKUP2 je dobivena udaljenost većinom ispod 100 000 metara i ne prelazi 200 00 metara. Za neuspješno geoparsirane zapise iz SKUP3 i SKUP4 može se uočiti kako su najveće dobivene udaljenosti niže u odnosu na najveće dobivene udaljenosti za neuspješno geoparsirane zapise korištenjem spaCy biblioteke. U ovom

slučaju, udaljenosti za neuspješno geoparsirane zapise iz SKUP2 ne prelaze 100 000 metara, a za SKUP3 ne prelaze 200 000 metara.

4.2.2. OpenAI uz GeoNames

U ovom potpoglavlju analizirat će se uspješnost iste metode kao i u potpoglavlju 4.2.1., ali koristeći drugi izvor podataka za pretraživanje prepoznatih lokacija u zapisima. Karakteristike rezultata korištenjem modela OpenAI-a uz GeoNames bazu prikazane su u sljedećoj tablici.

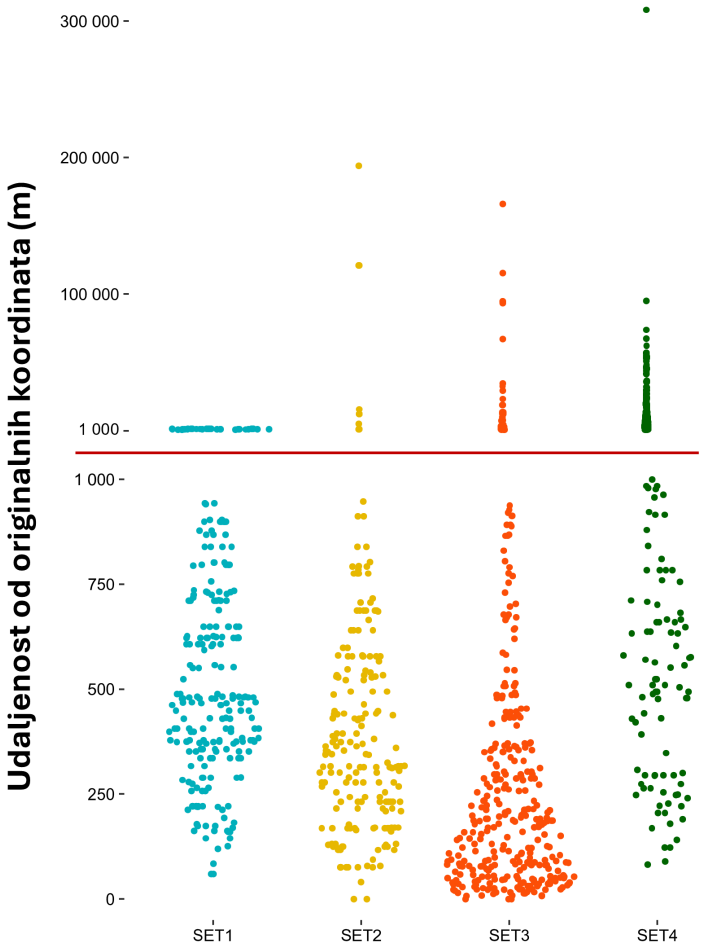
Korišteni skup podataka	SKUP1	SKUP2	SKUP3	SKUP4
Ukupno zapisa	246	224	389	341
Postotak geoparsiranih zapisa	91%	92%	94%	97%
Postotak uspješno geoparsiranih zapisa	87%	85%	82%	27%
Minimalna udaljenost (m)	60	0	0	83
Maksimalna udaljenost (m)	1,584	194,119	166,179	166,179
Prosječna udaljenost (m)	598	3,387	2,765	11,537

Tablica 4.7. Rezultati geoparsiranja

Kako se koristi ista metoda za prepoznavanje imenovanih entiteta kao i u potpoglavlju 4.2.1., indeks se pretraživao istim upitima. Postotak geoparsiranih zapisa je u odnosu na Registar geografskih imena manji za tri od četiri testna skupa podataka. Za SKUP1 postotak geoparsiranih zapisa niži je za čak 8.5% (21 zapis), za SKUP3 razlika je 2.9% (11 zapisa), a za SKUP4 1.2% (4 zapisa). Jedino je za SKUP2 postotak geoparsiranih zapisa ostao nepromijenjen. Postotak uspješno geoparsiranih zapisa također je manji za SKUP1 (za 5.3%), SKUP3 (za 7.2%) i SKUP4 (za 22.9%), ali za SKUP2 to nije slučaj. Za SKUP2 je došlo do povećanja od 3.2%. Kako se isto ponašanje, ali uz različite postotke, pojavljuje i prilikom usporedbe korištenja spaCy biblioteke na oba indeksa (opisano u potpoglavlju 4.2.2.), sada sa sigurnošću možemo reći kako je Registar geografskih imena povoljniji za geoparsiranje navedenih skupova.

Kako bi dobili jasniju sliku, usporedit ćemo rezultate prikazane u tablici 4.7. sa re-

zultatima prikazanim u tablici 4.4. gdje je korišten isti indeks, ali je za prepoznavanje imenovanih entiteta korištena spaCy biblioteka. Za SKUP1 postotak geoparsiranih podataka ostao je nepromijenjen, za SKUP2 on je smanjen za 6.4%, a za SKUP3 smanjen je za neznatnih 0.5%. Jedino povećanje može se uočiti za SKUP4 gdje je postotak geoparsiranih zapisa povećan za 6.8%. Analiziranjem broja uspješno geoparsiranih zapisa može se vidjeti kako su rezultati identični za SKUP1, dok se za SKUP2 broj uspješno geoparsiranih rezultata smanjio za 6.4%. Za SKUP3 broj uspješno geoparsiranih rezultata povećao se za 2.6%, dok povećanje za SKUP4 iznosi čak 18.2%. Iako se povećanje za SKUP4 čini značajnim, radi se o 14 zapisa koji su jako utjecali na postotak zbog općenito male uspješnosti na ovom skupu.



Slika 4.4. Prikaz udaljenosti od originalnih kooordinata za sve geoparsirane zapise

Prikaz udaljenosti za uspješno geoparsirane zapise korištenjem modela OpenAI-a ponovno skoro identično odgovara prikazu udaljenosti za uspješno geoparsirane zapise ko-

rištenjem spaCy biblioteke sa slike 4.2. Udaljenosti za SKUP3 opet su najviše grupirane oko nule, ali puno raspršenije po osi ordinata u odnosu na sliku 4.3., te je prosječna udaljenost uspješno geoparsiranih zapisa za SKUP3 i ovdje manja od prosječne udaljenosti geoparsiranih zapisa za SKUP1, nominalno najuspješnijeg skupa. Kod neuspješno geoparsiranih zapisa, jedina veća razlika u odnosu na sliku 4.2. je vidljiva za SKUP4. Udaljenosti od originalnih koordinata za skoro sve neuspješno geoparsirane zapise za SKUP4 su sada ispod 100 000 metara. Dodatno, za samo jedan zapis je dobivena udaljenost iznad 100 000 metara dok je prilikom korištenja spaCy biblioteke broj takvih zapisa bio znatno veći.

4.3. Mordecai

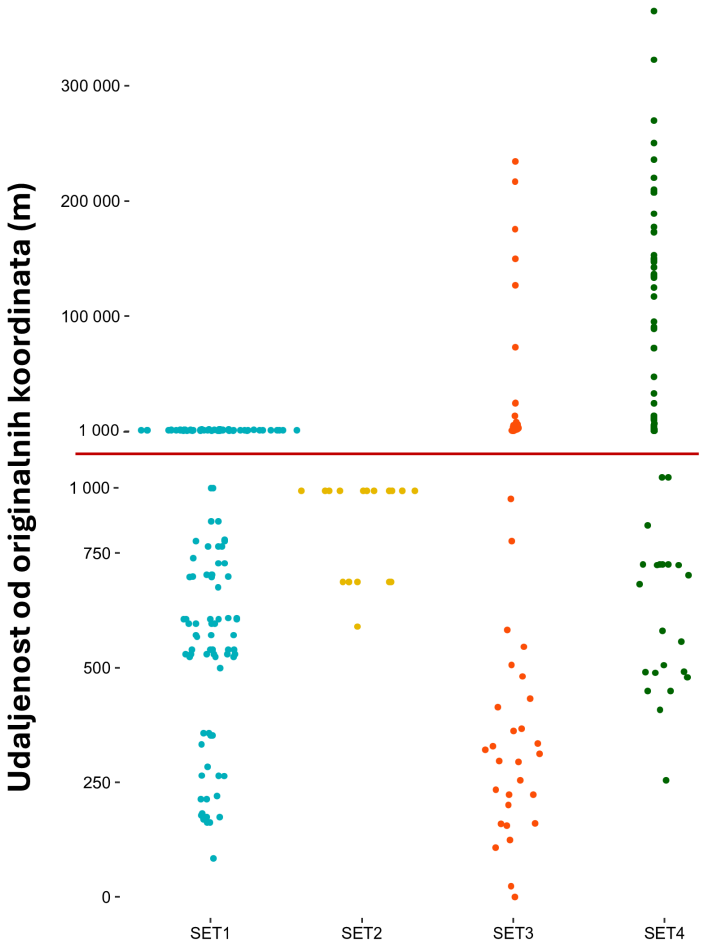
U ovom poglavlju analizirat će se uspješnost mordecai biblioteke na istim testnim skupovima podataka. Karakteristike rezultata korištenjem mordecai biblioteke prikazane su u sljedećoj tablici.

Korišteni skup podataka	SKUP1	SKUP2	SKUP3	SKUP4
Ukupno zapisa	246	224	389	341
Postotak geoparsiranih zapisa	53%	9%	16%	33%
Postotak uspješno geoparsiranih zapisa	30%	9%	7%	7%
Minimalna udaljenost (m)	85	590	0	255
Maksimalna udaljenost (m)	2,168	886	166,179	365,062
Prosječna udaljenost (m)	897	812	2,765	68,863

Tablica 4.8. Rezultati geoparsiranja

Iz tablice 4.8. jasno se može vidjeti kako je ova biblioteka, specifično razvijena za geoparsiranje, dala izrazito loše rezultate. To posebno dolazi do izražaja kada se ti rezultati usporede s rezultatima opisanima u prethodnim poglavljima. Nekoliko je razloga za ovako loše rezultate. Prvi je svakako interno (od strane mordecai biblioteke) korištenje spaCy biblioteke koja ne radi dobro na kratkim frazama. Za mordecai nije tako jednostavno utjecati na rezultate koje vraća spaCy biblioteka i u ovom radu se to nije dorađivalo,

nego se utjecalo na rezultate spaCy biblioteke samo onda kada se ona samostalno koristila. Drugi razlog je nemogućnost korištenja modela za hrvatski jezik u spaCy biblioteci. To znači da se svaki zapis trebao prevoditi na engleski, što je naravno podležno pogreškama, pogotovo uzevši u obzir da je i za ovo korišten model OpenAI API-a. Treći razlog je zadano korištenje GeoNames baze koja se kroz prethodne metode pokazala manje uspješna nego Registar geografskih imena. Na ovaj dio bi se moglo utjecati, ali za to je potrebno doraditi podatke koji se nalaze u Registru geografskih imena što će detaljnije biti opisano u poglavlju 5. Četvrti, a možda i najvažniji razlog je nemogućnost korištenja najnovije verzije spaCy biblioteke, što je detaljnije već objašnjeno u poglavlju 3.



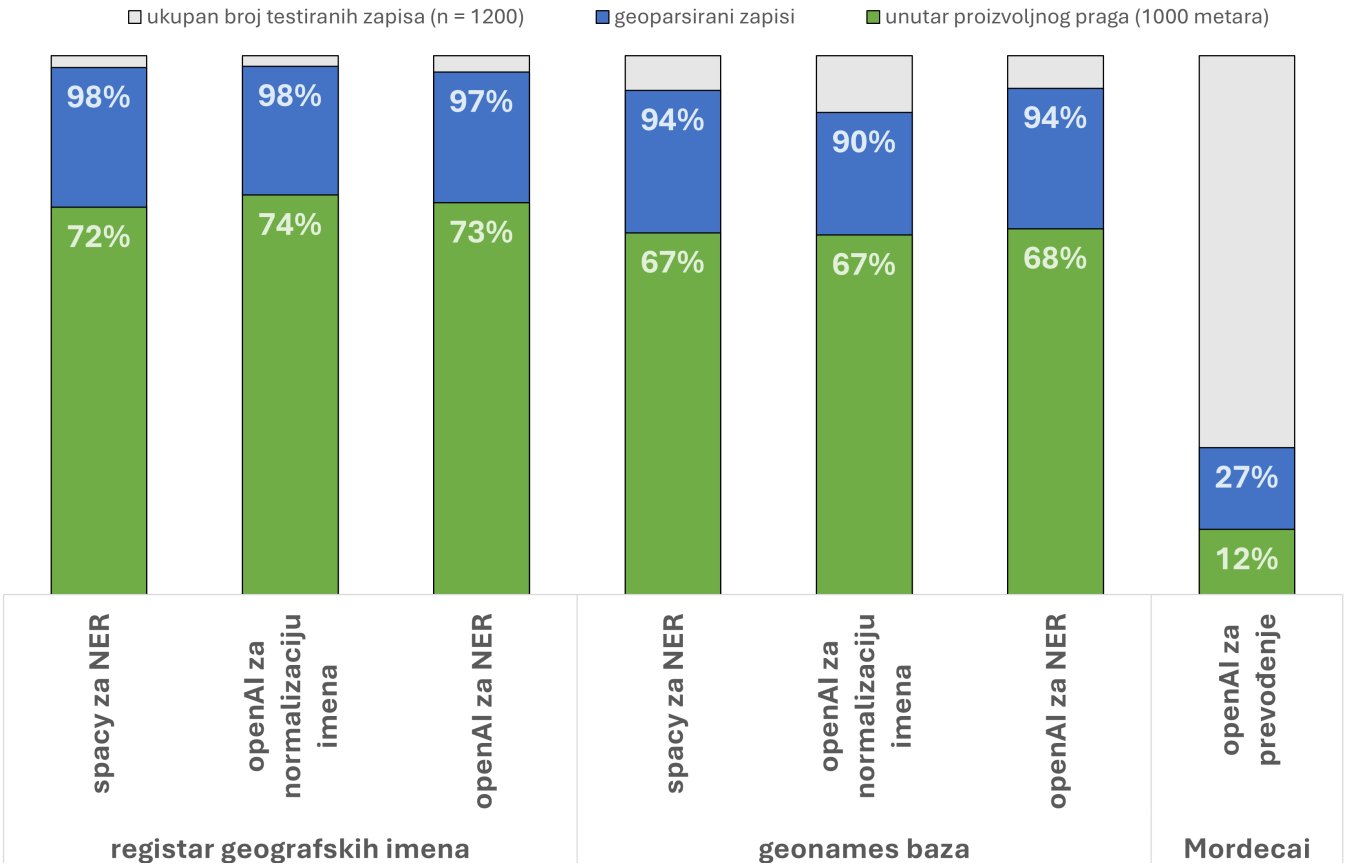
Slika 4.5. Prikaz udaljenosti od originalnih kooordinata za sve geoparsirane zapise

Na slici 4.5. jasno se može vidjeti razlika u broju geoparsiranih zapisa u odnosu na druge, prethodno opisane, metode. Iako je značajno manji broj zapisa geoparsiran, na skupovima SKUP1, SKUP3 i SKUP4 može se primjetiti slična distribucija udaljenosti kao

i prilikom korištenja ostalih metoda. Za SKUP2 je iznimno mali broj zapisa geoparsiran pa iz tog razloga nema smisla promatrati pripadnu distribucija udaljenosti.

4.4. Zaključak testiranja

Može se zaključiti kako je uspješnost geoparsiranja uvelike varirala ovisno o skupu podataka. Viši postotak uspjeha geoparsiranja zabilježen je u skupovima podataka koji su sadržavali detaljnije opise (više toponima u opisu lokacije), ali i koji su imali pravopisno ispravne zapise (SKUP1).



Slika 4.6. Sumarni graf testiranja

Na slici 4.6. može se vidjeti kako je na korištenim skupovima podataka najuspješnije bilo korištenje spaCy biblioteke uz normalizaciju te pretraživanjem Registra geografskih imena. To je pomalo i očekivano, izuzmemo li Mordecai, jer je spaCy biblioteka razvijena specifično za prepoznavanje imenovanih entiteta, no zbog složenosti hrvatskog jezika, za uspješnije pretraživanje indeksa, potrebno je normalizirati prepoznate entitete.

Iz slike 4.6. se također može vidjeti kako je geoparsiranje bilo uspješnije koristeći Registar geografskih imena u odnosu na GeoNames bazu. To nije iznenađujuće s obzirom na činjenicu da je Registar geografskih imena prilagođen za Hrvatsku te sadrži više zapisa za Hrvatsku nego GeoNames. Pomalo iznenađujući rezultat je korištenje modela OpenAI-a za prepoznavanje imenovanih entiteta. Uzevši u obzir da je korištenje spaCy biblioteke dorađeno kako bi radilo za kratke zapise, OpenAI bolje je odradio prepoznavanje imenovanih entiteta. Čak i uz dorade za spaCy biblioteku, iz rezultata vidljivih na slici, ne može se reći da je korištenje OpenAI-a dalo lošije rezultate u odnosu na korištenje spaCy biblioteke.

Metoda X	Broj uspješno geoparsiranih zapisa	Broj zapisa koje je samo metoda X uspješno geoparsirala
SpaCy uz Registar geografskih imena	863	2
SpaCy uz Registar geografskih imena i normalizaciju	890	25
SpaCy uz GeoNames	806	0
SpaCy uz GeoNames i normalizaciju	801	3
OpenAI uz Registar geografskih imena	873	7
OpenAI uz GeoNames	815	5
Mordecai	145	2

Tablica 4.9. Detaljnija analiza uspješno geoparsiranih zapisa

Kako se može vidjeti na slici 4.6. ukupan broj zapisa je 1200, a od njih čak 208 ni jedna od implementiranih metoda nije uspjela uspješno geoparsirati. Kakvo je već opisano, najuspješnija metoda je korištenje spaCy biblioteke uz normalizaciju te pretraživanje Registra geografskih imena koja je uspješno geoparsirala 890 zapisa. Od tih 890 zapisa, njih čak 25 niti jedna druga metoda nije uspjela uspješno geoparsirati te sada sa sigurnošću može reći kako je ovo najuspješnija metoda te se ona iz tog razloga postavlja kao zadana metoda geoparsiranja u ovom sustavu. Zanimljivo je primijetiti kako je korištenje spaCy biblioteke uz pretraživanje indeksa koji sadrži podatke iz GeoNames baze

jedina metoda koja nema niti jedan uspješno geoparsirani zapis koji je samo ta metoda uspješno geoparsirala.

Kako podaci, u pravilu, nisu savršeni, ni skupovi podataka korišteni u ovom radu nisu iznimka. Na primjer, u tablici 4.1. se može vidjeti kako je u SKUP2 korišten WGS84 sustav, ali su koordinate zaokružene na samo dvije decimale. S obzirom da se Hrvatska prostire na otprilike šest stupnjeva geografske dužine i četiri stupnja geografske širine, korištenje manje preciznih zapisa uvelike utječe na pogrešku. Tako pogreška od 0.01 stupanj za geografsku širinu i dužinu rezultira ukupnom pogreškom od približno 1,350 metara.

Isto tako u skupovima se mogu pronaći slični zapisi koji očekuju različite rezultate. Tako na primjer u SKUP1 toponimi "Industrijska ulica , Dugo Selo" i "Industrijska ulica - ulaz u Velebit promet , Dugo Selo" očekuju rezultate koji se razlikuju za 222 metra. Nama ljudima to je vrlo jednostavno za protumačiti, drugi toponim je nama specifičniji, no razvijeni sustav to ne razumije. Problem možda i ne bi bio toliko velik kada bi Registar geografskih imena sadržavao navedeni "Velebit promet" u toj ulici, ali kako baza nije zamišljena za takve toponime, ona ga ni ne sadrži pa to dovodi do pogreške.

Također je važno napomenuti kako skupovi podataka korišteni u ovom testiranju nisu provjereni te postoji mogućnost pogreške prilikom geokodiranja toponima u samim skupovima. To bi moglo dovesti do velikih pogrešaka, pogotovo ako su greške napravljene u zapisima koji koriste WGS84 referentni koordinatni sustav.

5. Moguća poboljšanja

Razvoj sustava gotovo nikad ne završava prvom verzijom. Ključ uspješnih sustava je u konstantnom održavanju i unapređivanju. Tako će se u ovom poglavlju opisati potencijalne nadogradnje koje bi rezultirale većom točnošću, ali i bržim izvođenjem samog procesa geoparsiranja.

5.1. Uvođenje *cache-a*

Najčešća radnja koja se izvodi u ovom sustavu je pretraživanje inkesa. Iako je ta operacija dosta brza (vrijeme izvođenja je manje od jedne stotinke), uvođenje *cache-a* zasigurno bi ubrzalo aplikaciju prilikom obrade velikih excel datoteka. Ako za tekstualni zapisi u datoteci složeni i sadrže više imenovanih entiteta, te ako excel datoteka sadrži mnogo zapisa (redaka), tada se baza treba pozvati za svaki prepoznati imenovani entitet. Kako se opažanja često provode na nekom užem geografskom području (na primjer grad Zagreb), vrlo je vjerojatno da će se isti imenovani entitet ponoviti u više zapisa. Samim time što je to isti imenovani entitet, na istom području, nema smisla ponovno pretraživati indeks jer će on vratiti iste rezultate, te bi u tom slučaju *cache* mogao uvelike pridonijeti performansama, a samim time i poboljšanju korisničkog iskustva.

5.2. Optimizacija normalizacije toponima

Za normalizaciju je u ovom radu korišten model "gpt-4o-mini" OpenAI API-a. Ovo nije najbolji dostupan model pa bi prvi korak bio isprobavanje modela "gpt-4o". Postotak uspješnno normaliziranih toponima vjerojatno bi bio točniji, ali to bi rezultiralo i dužim čekanjem odgovora, koje je već sad u minutama. Bolje rješenje bilo bi razvijanje vlastitog modela koji je treniran specifično za to, pretvaranje riječi u nominativ, a zatim i korištenje tog modela u ovom sustavu. To bi sigurno rezultiralo kraćim vremenom izvo-

đenja, a dobrim treniranjem modela zasigurno bi se povečala i točnost. Još jedan način na koji se ovo može pokušati riješiti je povećanje *fuzziness* atributa prilikom pretrage indeksa. Jedini problem kod ovog pristupa je što se treba napraviti detaljna analiza koliki je *fuzziness* prihvatljiv što zasigurno nije jednostavan zadatak. Preveliko povećanje tog atributa moglo bi rezultirati lažnim pozitivima zbog sličnosti imena nekih toponima, što bi direktno utjecalo i na uspješnost geoparsiranja.

5.3. Uvođenje obrade prirodnog jezika

Prilikom zapisivanja opažanja često se lokacija opisuje prema blizini nekog poznatijeg toponima. Tako na primjer ako primjetimo neku zanimljivu pojavu ili životinju u blizini jezera Jarun najvjerojatnije ćemo napisati "200 metara sjeverno od jezera Jarun". U tom slučaju će razvijeni sustav prepoznati imenovani entitet Jarun te vratiti njegovu geolokaciju, ali izgubit će se informacija koja nam preciznije opisuje lokaciju. Taj problem mogao bi se riješiti uvođenjem modela za obradu prirodnog jezika. Tom modelu dao bi se čitav tekst zapažanja, a njegov zadatak bi bio da lokaciju toponima, koju sada sustav vraća, promjeni na način da odgovara opisu u tekstu. Odnosno, model bi iz konteksta trebao prepoznati sadrži li tekst usporedbu s nekim toponimom, te ako sadrži trebao bi promijeniti lokaciju samo tog toponima. Na taj način ne bi bilo gubitka informacija, a sustav bi za svakodnevne zapise davao točnije rezultate. Ovaj problem je već primijećen te je detaljnije razrađen u radu [13].

5.4. Korištenje Registra geografskih imena u mordecaiu

Kako je Registar geografskih imena rezultirao većim postotkom uspješno geoparsiranih zapisa u odnosu na GeoNames bazu, trebalo bi ga pokušati koristiti u mordecai biblioteci. Za to je potrebno napraviti dorade nad podatcima koji su potrebni mordecaiu (npr. alternativna imena toponima, regija, država, populacija...). Naime mordecai se oslanja na potpis dokumenta iz GeoNames baze, te koristi razne atribute koje ti dokumenti sadrže kako bi odredio najboljeg kandidata za rezultat geoparsiranja. Ti podaci trenutno ne postoje za dokumente u Registaru geografskih imena te iako bi njihovo dodavanje moglo biti vrlo dugotrajan i mukotrpan proces, zasigurno bi rezultirao većom uspješnosti mordecai biblioteke.

6. Zaključak

Kako bi se omogućila automatizirana procjena prostorne točnosti skupova podataka, u ovom radu razvijena je aplikacija koja koristi različite metode za proces geoparsiranja. Metode su razvijene korištenjem spaCy i mordecai biblioteka te korištenjem OpenAI API-a. Prilikom testiranja korištene su dvije baze podataka, Registar geografskih imena i GeoNames. U radu su prikazani rezultati testiranja svih razvijenih metoda uz obje baze te su ti rezultati međusobno uspoređeni. Iako neke metode pokazuju visoku uspješnost geoparsiranja, od preko 90%, uz vrlo strogi predodređeni prag, sustav itekako ima mjesta za napredak. Kako je aplikcija razvijena na klijent-poslužitelj arhitekturi, nadograđivanje ovih metoda, kao i potencijalno dodavanje novih metoda, ne bi trebao biti težak zadatak. U radu su također predložena moguća unaprijeđenja aplikacije koja mogu biti temelj za daljnje istraživanje.

Literatura

[1] D. Šlamon, F. Varga, i A. K. c Divjak, “Towards development of a tool for the automated assessment of the spatial accuracy of nature observation datasets”, 2023. [Mrežno]. Adresa: https://www.croris.hr/crosbi/publikacija/prilog-skup/732119

[2] https://docs.docker.com/get-started/docker-overview, [mrežno; stranica posjećena: siječanj 2025.].

[3] https://phoenixnap.com/kb/what-is-docker, [mrežno; stranica posjećena: siječanj 2025.].

[4] A. Halterman, “Mordecai 3: A neural geoparser and event geocoder”, *arXiv preprint arXiv:2303.13675*, 2023.

[5] “Geoparser - github repozitorij koji sadrži kod razvijen u sklopu ovog rada”. [Mrežno]. Adresa: https://github.com/ipavic-fer/GeoParser

[6] “Es geonames”. [Mrežno]. Adresa: https://github.com/openeventdata/es-geonames

[7] “Geoportal - nipp”. [Mrežno]. Adresa: https://geoportal.nipp.hr/geonetwork/srv/hrv/catalog.search#

[8] D. Vlahović, “Invazivna flora zagrebačke županije - biogeografija i potencijalno širenje”, doktorska disertacija, Prirodoslovno-matematički fakultet, 2018., dostupno na: https://dr.nsk.hr/islandora/object/pmf%3A3162.

[9] “Cpgrd - croatian plant genetic resources database”. [Mrežno]. Adresa: https://cpgrd.hapih.hr/gb/map/main/index/0

[10] H. V. Gašparić, K. M. Mikac, I. P. Živković, B. Krehula, M. Orešković, M. A. Galešić, P. Ninčević, F. Varga, i D. Lemić, "Firefly occurrences in croatia—one step closer from citizen science to open data", *Interdisciplinary Description of Complex Systems*, 2022. [Mrežno]. Adresa: https://indecs.eu/index.php?s=x&y=2022&p=112-124

[11] D. Pavić, D. Grbin, S. Hudina, U. P. Zmrzljak, A. Miljanović, R. Košir, F. Varga, J. Ćurko, Z. Marčić, i A. Bielen, "Tracing the oomycete pathogen saprolegnia parasitica in wild fish populations: challenges and future directions", *Scientific Reports*, 2022. [Mrežno]. Adresa: https://www.nature.com/articles/s41598-022-16553-0

[12] "Geonames - geografske informacije svijeta". [Mrežno]. Adresa: http://www.geonames.org

[13] M. A. SYED, M. Roche, i M. Teisseire, "Geospacy: A tool for extraction and geographical referencing of spatial expressions in textual data", str. 115–126.

Sažetak

Usluga za automatiziranu procjenu prostorne točnosti skupova podataka opažanja u prirodi

Zbog nedostatka standardiziranog protokola prikupljanja geoprostornih podataka iz opažanja pojavila se potreba za automatiziranom procjenom točnosti prikupljenih podataka. U sklopu ovog rada razvijena je aplikacija koja koristi različite metodu u tu svrhu. Opisane su implementacije tih metoda kao i rezultati testiranja aplikacije. Na kraju su predloženi i daljnji koraci za moguće unaprijeđenje procesa procjene točnosti geoprostornih podataka.

Ključne riječi: Geoparsiranje; Mordecai; SpaCy; Elasticsearch; Python

Abstract

Usluga za automatiziranu procjenu prostorne točnosti skupova podataka opažanja u prirodi

Due to the lack of a standardized protocol for collecting geospatial data from observations, there was a need for an automated assessment of the accuracy of the collected data. As part of this paper, an application was developed that uses different methods for this purpose. The implementation of these methods, as well as the results of application testing, are described. Finally, further steps for potential improvements in the geospatial data accuracy assessment process are proposed.

Keywords: Geoparsing; Mordecai; SpaCy; Elasticsearch; Python