TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Jakob Pindis 183213IAPM

PILTIDE PEALKIRJADE JA KOORDINAATIDE ALUSEL TURISMIOBJEKTIDE KOONDAMINE JA TÜÜBI KATEGOORIATE MÄÄRAMINE NING NEID TOETAV PAINDLIK RAAMISTIK

Magistritöö

Juhendaja: Ago Luberg
MSc

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Jakob Pindis

01.05.2020

Annotatsioon

Käesoleva töö eesmärgiks oli välja töötada Sightsmap rakenduse tarbeks raamistik, mis võimaldab teha katsetusi ja saada kiiret tagasisidet proovitud turismiobjektide kategoriseerimise algoritmidest ja masinõppe mudelitest. Praeguse hetkeni puudus vahend Sightsmap rakenduse spetsiifiliste katsetuste tegemiseks.

Suur osa tööst oli leida ning proovida erinevaid algoritme ja masinõppe mudeleid kinnitamaks loodava rakenduse kasutatavust. Ühtlasi on lisatud algoritmid näidisteks kasutajatele rakenduse hõlpsamaks kasutamiseks ja laiendamiseks tulevikus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 34 leheküljel, 7 peatükki, 7 joonist, 2 tabelit.

Abstract

Supportive framework for tourism objects clustering and determination of the type categories using photo titles and coordinates

The purpose of this master thesis was to develop photo data categorisation experimenting framework specifically for Sightsmap application. Until now there was no common way for rapidly testing new categorisation algorithms and machine learning models for Sightsmap tourism objects.

First step of work was to evaluate at hand Panoramio photo data and figure out application architecture. It was decided that solution would be to build four separate modules: machine learning, API server, database and frontend module.

After that, research about different sample algorithms and models started. Different algorithms and pretrained, self-trained models were experimented with. Options that worked decently were added to machine learning module. Added options also serve as examples to help future application users to add their own implementations or improve existing ones.

During algorithms and models experimenting period, plug and play, rapid categorisation goodness testing support needed to be added as well. All algorithms, models that were added by author and future algorithms, models that will be added, are testable using pre-set test data. Test data can simply be extended by providing more test samples.

There were many problems on the way that needed to be solved. To highlight few, initially mapping out whole process that worked considering it needed to function with quite large dataset and finding different algorithms and models that would work with provided dataset.

The thesis is in Estonian and contains 34 pages of text, 7 chapters, 7 figures, 2 tables.

Lühendite ja mõistete sõnastik

API	*Application programming interface*, rakendusliides
Backend	Rakenduse serveripoolne osa
Crowdsourcing	Ühisloome, tavaliselt suur hulk vabatahtlike inimesi kes panustavad teatud probleemi lahendamiseks
CSV	*Comma-separated values*, fail, kus kirje väärtused on eraldatud komaga ja iga kirje on uuel real
Frontend	Rakenduse kliendipoolne osa
GeoJSON	JSON formaat, mis on mõeldud geograafiliste andmete edastamiseks
HTTP	Protokoll teabe edastamiseks arvutivõrkudes
JSON	*JavaScript Object Notation*, formaat andmete edastamiseks, mis sisaldab teksti kujul järjendeid ning atribuut-väärtus paare
Kategoriseerimine	Rühmitamine sarnaste tunnuste alusel
Kogum	Kompleks, hulk esemeid või nähtusi ühtse rühma või tervikuna, antud töö raames tähistab kogum turismiobjekti, mille ümbruses on pildid tehtud
Koondamine	Hajali-, laialiolekust ühte kohta kokku või hulgaks koguma
Lemma	Sõna algkuju
Mudel	Keeruka objekti, algoritmi käitumine
REST	*Representational State Transfer*, tarkvaraarhitektuuri stiil veebiteenuste loomiseks
Stopwords	Nimekiri tihti esinevatest sõnadest mida ignoreerida
Word embedding	Sõnade kaardistamine vektoritena

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Töö on seotud Sightsmap [1] rakenduse tarbeks loodava lisarakendusega, et paremini toetada sealset arendust ja funktsionaalsuse laiendamist. Sightsmap on veebirakendus, mis kuvab soojuskaardi abil vaatamisväärsuste populaarsust üle maailma. Sightsmapi peamiseks andmeallikaks on Panoramio *crowdsourcing* pildiandmed, kust on leitud turistidele huvipakkuvad vaatamisväärsused, mille kohta soojuskaart on genereeritud. Üks tulevikuplaanidest Sightsmapis on soovitada vaatamisväärsusi turistidele kogu maailmas. Soovitused peaksid põhinema turistide huvidel, selleks on vaja määrata kõigile vaatamisväärsuste objektidele sobivad kategooriad.

Praeguse hetkeni ei olnud võimalust uusi turismiobjektide kategoriseerimise ideid kiiresti katsetada ning teha seda ühes kohas ühtse stiili alusel. Käesoleva töö eesmärgid on töötava raamistiku loomine koos mudelite ja testidega, mida on serveris lihtne juurutada.

Olulisemad sammud mida rakendus peab suutma täita:

- piltide pealkirjade ja koordinaatide alusel ruumipunktide koondamine kogumiteks
- kogumite pealkirjadest kategooriate leidmine
- kogumite kategoriseerimine
- saadud tulemuste kiire hindamine ning visualiseerimine

Rakenduse sisendandmete osas on võimalik kasutada erinevaid allikaid, mille põhjal saab leida huvipakkuvaid turismiobjekte. Antud töö skoobis on see piiratud Panoramio Briti saarte *crowdsourcing* pildiandmetega. Lisaks teiste riikide Panoramio pildiandmetele saab tulevikus laiendada kategoriseerimise funktsionaalsust näiteks kasutades sotisaalmeedia postitusi koos asukohaga [2].

Loodud lahendus koosneb neljast eraldiseisvast moodulist:

- masinõpe – kõige mahukam osa, sisaldab mudeleid ja tulemuste hindamist testide alusel
- veebirakendus – tulemuste visuaalne hindamine
- API server – liides veebirakenduse ja andmebaasi vahel
- andmebaas – masinõppe mooduli tulemuste salvestamiseks

Loodud rakenduse kood on nähtav GitLab repositooriumis[1].

[1] https://gitlab.cs.ttu.ee/Jakob.Pindis/photodataapps

2 Taust ja seotud tööd

Uurides tausta, siis oli aru saada, et teema on aktuaalne ja on erinevate vaatenurkade alt uuritud. Aktuaalsust tõstab üha suurenev seadmete hulk, mis suudavad edastada teksti koos asukohainfoga. Järgnevalt on kirjeldatud, mis on erinevused varasemalt tehtud töödega, mis on töös keerulist ja milles seisneb uudsus.

Artiklid [3], [4] annavad alguspunkti, millest alustada tausta uurimist. Nendest saab üldiselt võtta kokku enimkasutatavad viisid, kuidas piltide tuvastamist on uuritud antud lõputöö kontekstis, kus märksõnadeks on turismiobjektid, pildid, pildiandmed, metaandmed. Artiklite alusel saab luua kolm lähenemissuunda, millel igaühel on omakorda alamsuunad. Esimene suund, millest räägitakse, on piltide pikslite abil kategoriseerimine. Seda varianti töös ei rakendata, sest olemas on ainult piltide pealkirjad teksti kujul ja koordinaadid. Teise suunana tuuakse välja piltide pikslite ja metaandmete kombinatsioonina kategoriseerimine, mis jääb samuti välja piltide olemasolu puudumise tõttu.

Kolmas ja kõige ligilähedasem suund, millel tasub pikemalt peatuda, on ainult piltide metaandmete alusel piltide kategoriseerimine. Metaandmete kasutamisel toetutakse tavaliselt siltidele (*tag*), mis on kasutaja pool varasemalt juba määratud. Näiteks kategoriseeritakse siltide abil või siltide ning ruumipunkte vaheliste seoste abil [5]. Panoramio pildiandmetel silte ei ole, seega otsiti edasi ainult tekstitöötlusega seotuid uurimistöid.

Kuna pildituvastus toimub pealkirja põhiselt, tuleb piltide pealkirjade tuvastamiseks kasutada loomuliku keele töötluse võtteid. Alustuseks katsetatakse lihtsamaid variante nagu sõna sageduse põhine tuvastus, hiljem TD-IDF [6], [7], *word embedding'*ut [8], [9], [10] mis on ülesehituselt keerulisemad, uuemad tehnoloogiad ja võivad anda paremaid tulemusi. Samas ei ole paremad tulemused garanteeritud igasuguste andmetega, mille pealt soovitakse tuvastada teksti sisulist tähendust [11].

Tekstitöötluse osa on hiljuti palju uuritud erinevate eesmärkide saavutamiseks. Keeruliseks teeb selle siiski asjaolu, et tegu on *crowdsourcing* andmetega, mis on vabas vormis pealkirjad, millel puudub tihtipeale kontekst, sest pealkirjad on napisõnalised või

üldised. Uurides taoliste andmete kategoriseerimise kohta, siis avaldub, et selliste tekstide hästi kategoriseerimine ei ole alati lihtne ülesanne [12].

Lisaks loetud kirjandusele otsiti olemasoleval ja kasutataval kujul praktilist rakendust, mis annaks soovitud tulemuse, aga sellist rakendust ei leitud. Põhjuseks on tõenäoliselt asjaolu, et tegu on Sightsmap rakenduse jaoks spetsiifilise lahendusega.

Töös on tarvis kokku ühendada mitmeid varem uuritud eraldiseisvaid osasid, mida on erinevate eesmärkide saavutamiseks uuritud, aga see ei tähenda, et saab võtta valmis osad ja need lihtsalt kokku panna. Selle kinnituseks saab tuua mõned näited. Piltide pealkirjad on tihtipeale ilma kontekstita, samal ajal tuleb võtta arvesse, et andmemahud on suured, mis seab piiri algoritmi/mudeli keerukusele. Probleemiks on ka mudelite treenimiseks vajalike treeningandmete puudumine. See näitab, et terviklahenduse loomine ei ole sirgejooneline ja iga osa vajab eraldi uurimist ja praktilist katsetamist, et leida mõistlikud variandid, mis saab ühendada kokku tervikuks.

Kuigi suurem rõhk töös on masinõppe moodulis teksti kategoriseerimise osal, siis ei saa alahinnata teisi lõputöös tehtud osi. Tuli tegeleda raamistiku arhitektuuri väljamõtlemisega, sisendandmete koondamisega, kategooriate genereerimisega dünaamiliselt kogu andmehulgalt ja lõpetuseks tulemuste headuse hindamise süsteemi loomisega, võttes aluseks *f-score* hinnangu mitme kategooria toega [13].

Töö uudsus seisneb selles, et selle tulemusena saab kiiresti hinnangu, kui hästi või halvasti erinevad algoritmid, isetreenitud ja eeltreenitud masinõppe mudelid antud töö iseloomust tulenevas kontekstis toimivad. Lisaks valmis rakendus, mida saab eesmärgipõhiselt kasutada ja mida tervikuna samal kujul varasemalt ei eksisteerinud.

3 Sisendandmed

Sisendandmetena saab kasutada erinevaid allikaid, kus on minimaalselt olemas pealkiri ja geograafilised koordinaadid. Lõputöö raames kasutati Panoramio pildiandmed. Need olid töö juhendajale suures mahus kättesaadavad.

3.1 Panoramio pildiandmed

Panoramio pildiandmed on CSV failis kujul id, laiuskraad, pikkuskraad, pildi id, autori id, autor, kuupäev, pealkiri. Kasutati Briti saarte Panoramio pildiandmeid, mis sisaldasid 1,41 miljonit rida. Nendest kasutati id, laiuskraadi, pikkuskraadi ja pealkirja väärtusi. Vajadusel saab tulevikus rakendada veel teisigi väärtusi. Samas mida rohkem spetsiifilisi andmeid kasutada, seda rohkem limiteerib see rakenduse paindlikkust, sest erinevatest allikatest andmetel ei pruugi olla kõiki vajalikke lisaväärtusi. Teisalt on rohkemate andmetega suurem tõenäosus saada paremaid tulemusi.

Tabel 1 abil on välja toodud ülevaade kasutatud Panoramio andmetest. Lisas 1 on näidatud lõik kõikide veergudega Panoramio CSV failist.

Tabel 1. Lõik Panoramio pildiandmetest kasutatud veergudega.

id	laiuskraad	pikkuskraad	pealkiri
1416501	50.923489	0.923023	Road Closed
1416502	50.918348	0.981772	Bri and his Bass
1416503	50.912612	0.975358	"New Lighthouse, Dungeness, Kent"
1416504	50.917740	0.980980	""
1416505	50.918337	0.981788	"My bait, well some."
1416506	50.901754	0.959879	Dungeness - new lighthouse
1416507	50.920577	0.976775	Great potential with excellent rail connection!
1416508	50.918338	0.981785	Nikki on the boat
1416509	50.944098	0.883123	Lydd Caravan Park
1416510	50.916517	0.977186	""
1416511	50.914435	0.969844	Dungeness railway station
1416512	50.916517	0.977186	sparrow
1416513	50.913854	0.969694	Dungeness railway station

4 Analüüs

Töö algusfaasis tuli aru saada olemasolevate andmete struktuurist ja rakendusele seatud soovidest. Seejärel sai paika panna nõuded ning loodava rakenduse arhitektuuri.

4.1 Funktsionaalsed nõuded

Funktsionaalsed nõuded on paika pandud Sightsmap arendaja seisukohast, kes saab teha järgnevaid tegevusi:

- saab lisada masinõppe moodulis uusi sisendandmete salvestamise algoritme
- saab lisada masinõppe moodulis uusi kogumitesse koondamise algoritme
- saab lisada masinõppe moodulis uusi kategooriate leidmise algoritme
- saab lisada masinõppe moodulis uusi kategoriseerimise algoritme
- saab lisada masinõppe moodulis juurde uusi või vahetada välja protsessi vahesammudena käivitatavaid algoritme
- saab lisada masinõppe moodulis juurde uusi või muuta olemasolevaid test- ja treeningandmeid
- saab võrrelda masinõppe moodulis erinevaid kategoriseerimise algoritmide tulemusi testide alusel
- saab käivitada masinõppe moodulis erinevate kategoriseerimise algoritmide testid kõik korraga või ühekaupa
- saab kasutada masinõppe moodulis testtabeleid kogu protsessi käivitamiseks eraldi ilma põhitabeleid muutmata
- saab masinõppe ja API serveri logimist suunata nii konsooli kui ka faili

- saab veebirakendusest vaadata eeltöötluseta salvestatud pildiandmeid põhitabelist ja testtabelist
- saab veebirakendusest vaadata töötlusprotsessi läbinud salvestatud pildiandmete lõpptulemust põhitabelite ja testtabelite alusel
- veebirakendus peab laadima valitavad kategooriad dünaamiliselt andmebaasist
- saab veebirakenduse kaudu otsida kohti asukoha nime järgi
- saab veebirakenduse kaudu piirata otsingut raadiuse põhiselt kilomeetrites
- saab veebirakenduse kaudu piirata kogumite otsingut kategooriate järgi
- saab veebirakenduse kaudu piirata kogumite otsingut populaarsuse järgi
- saab masinõppe, API serveri ja veebirakenduse moodulites hoida eraldi arendamise ja serveri juurutamise jaoks keskkonnapõhist konfiguratsiooni

4.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded on järgmised:

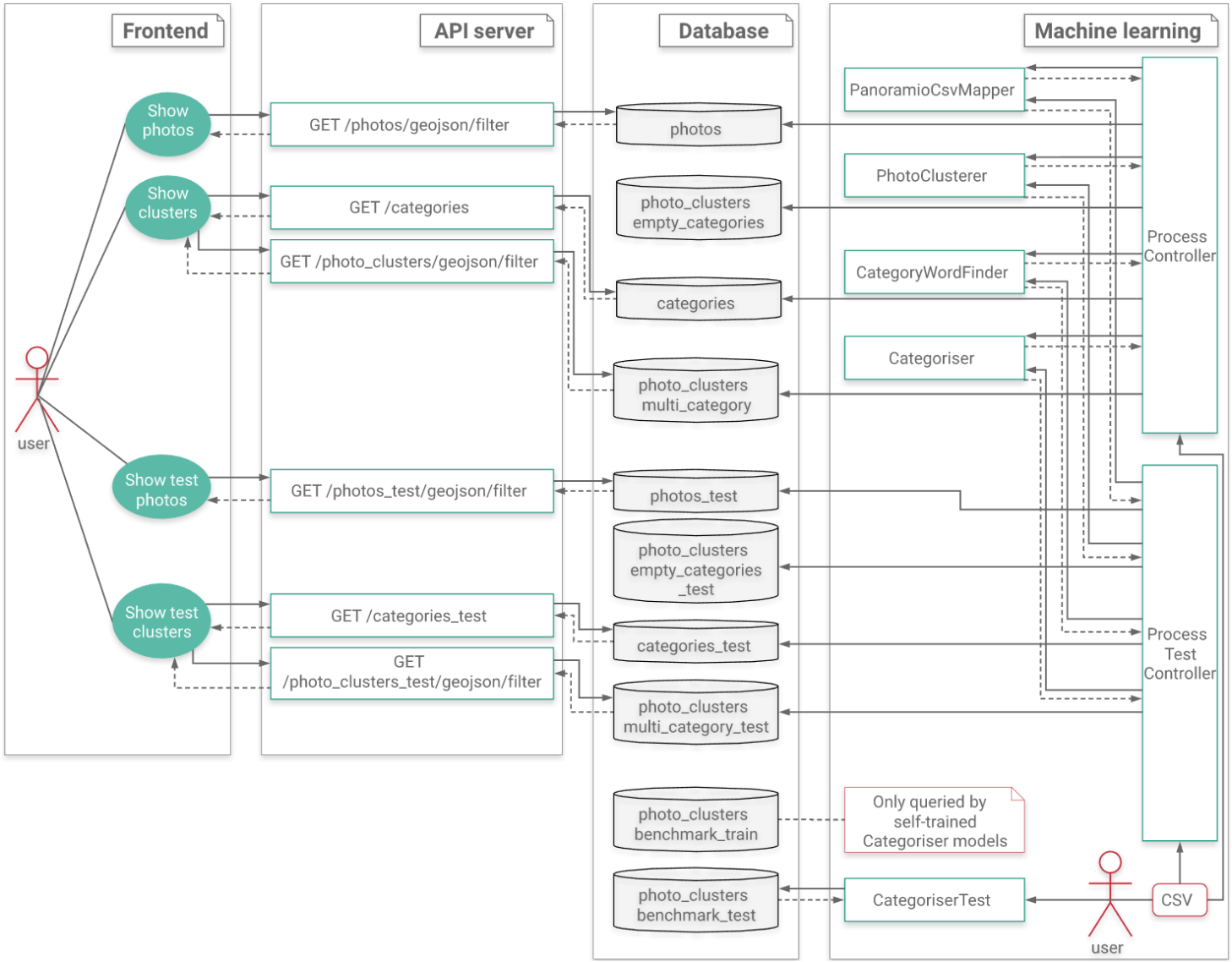
- rakendus on jaotatud eraldiseisvateks sõltumatuteks mooduliteks
- rakenduses on etteantud mustrid või näited mis lihtsustab uute algoritmide või mudelite lisamist
- rakendus peab olema väheste sammudega võimalik juurutada serveris

4.3 Arhitektuur ja tehnoloogia valik

Projekt on arhitektuuriliselt jaotatud neljaks mooduliks [14]. Moodulid on Dockeri[1] konteinerites [15]. Mooduleid on võimalik üksteisest sõltumata arendada ja vajadusel täielikult välja vahetada.

[1] https://www.docker.com

Järgnevalt on Joonis 1 abil ülevaatlikult näidatud arhitektuur ja tegevuste protsess. Igal tabelil, välja arvatud tulemuste otseseks testimiseks mõeldud *benchmark* tabelitel, on olemas testtabelid. Nende idee seisneb selles, et kui põhitabelites soovitakse hoida teatud salvestatud seisu alles, mis võib olla mahult väga suur, siis testtabelid võimaldavad teha paralleelselt katsetusi ilma põhitabelite tulemusi eraldi ringi salvestamata või kustutamata. See võimaldab töötada paindlikumalt ja kiiremini.



Joonis 1. Üldine rakenduse arhitektuur.

Arhitektuuriliselt on *backend* osas kaks Pythoni[1] programmeerimiskeeles realiseeritud moodulit, nendeks on masinõppe ja API server moodulid. *Frontend* osas on üks veebirakenduse moodul, mis on realiseeritud JavaScriptis[2]. Neljandaks mooduliks on PostgreSQL andmebaas.

Pythoni teekide sõltuvusi hallatakse Poetry[3] abil. See võimaldab *backend* moodulites vajadusel lisada ja samas jälgida olemasolevaid sõltuvusi Poetry konfiguratsioonifaili abil. JavaScripti teekide sõltuvusi hallatakse NPM[4] abil. Kõik moodulid, välja arvatud andmebaas, võimaldavad määrata konfiguratsioonifailis arendamise ja serveris juurutamise keskkondadele erinevad parameetrid, mis lihtsustab arenduse ja juurutamise protsessi serveris. Masinõppe ja API server moodulites on võimalik konfigureerida logimist. Logimisest on täpsemalt kirjutatud Lisas 2.

4.3.1 Masinõppe moodul

Masinõppe mooduli ülesanneteks on teostada järgmisi samme: CSV failist andmete lugemine, CSV failist loetud pildiandmete koondamine kogumiteks, kategooriate leidmine, kogumite kategoriseerimine, kategoriseeritud kogumite andmebaasi salvestamine ja tulemuste hindamine. Tegu on mooduliga, kuhu on võimalik lisada juurde uusi algoritme, mudeleid ja neid testida. *Controller* klasside abil on võimalik erinevaid protsessi samme sisse/välja lülitada, asendada, järjestust muuta või lisada uusi. Lihtsamaks teeb uute algoritmide ja mudelite lisamise see, et ette on lisatud näidised, mis kujul peaks uus lisatud algoritm või mudel väljundi andma. Kui suudetakse luua samal kujul väljund, saab selle salvestada andmebaasi, nagu iga olemasolev algoritm või mudel seda teeb.

Tulemuste hindamise üks pool on realiseeritud masinõppe moodulis ja teine veebirakenduse moodulis. Masinõppe moodulisse on lisatud testidepõhine hindamine, mis on realiseeritud Pythoni unittest[5] raamistiku abil. Lisatud on testimise näited, mida saab taaskord võtta aluseks, kuhu lisada järjest uusi testitavaid algoritme ning mudeleid

[1] https://www.python.org

[2] https://developer.mozilla.org/en-US/docs/Web/JavaScript

[3] https://python-poetry.org

[4] https://www.npmjs.com

[5] https://docs.python.org/3.6/library/unittest.html

juurde. Uute testide lisamine on üritatud teha võimalikult lihtsaks. Testid on eraldi kirjas testtabelis, kus iga rida on uus test. Kui tulevikus peaks tekkima ressurssi, et kirjeldada ära olulisemalt suuremal määral piltide kogumite kategooriaid, siis piisab sellest, kui lisada andmebaasi tabelisse uus rida juurde. Testimine annab ligikaudse, aga kiire hinnangu lisatud algoritmi või mudeli headusest.

Testide tabeli kõrvale on loodud treeningandmete tabel. Seda tabelit kasutatakse isetreenitud mudelite treenimiseks. Sarnaselt testtabeliga, saab treeningandmete tabelisse lisada andmebaasi ridu ehk treeningandmeid juurde, mis võimaldab luua paremaid isetreenitud mudeleid.

4.3.2 Andmebaasi moodul

Andmebaasi moodulit kasutatakse masinõppe mooduli poolt loodud andmete salvestamiseks ja API server mooduli poolelt andmete küsimiseks. Andmebaasina kasutatakse PostgreSQL[1] andmebaasi PostGIS[2] laiendiga. Moodul on võimalik käivitada Dockeri konteinerina, mis loob käivitamise hetkel vajalikud tabelid ning täidab testimiseks ja treenimiseks mõeldud tabelid andmetega.

Algselt plaaniti rakendada PostGIS laienduse raadiuse arvutamise funktsionaalsust masinõppe moodulis Panoramio pildiandmete koondamise etapis. Selgus, et see osutus liiga aeglaseks, selleks loodi masinõppe moodulis kohandatud lahendus, koondamise kohta saab täpsemalt lugeda alampunktis 5.2. PostGIS laiendi raadiuse funktsionaalsust sai siiski rakendada API serveri moodulis, kus intensiivset protsessimist ei toimu ja kiirus probleemiks ei olnud.

4.3.3 API server moodul

API server moodul võimaldab küsida andmebaasist vajalikke andmeid. Moodul on mõeldud kasutamiseks veebirakenduse poolt liidesena andmebaasi vahel. Liiklus veebiserveri ja API serveri vahel käib üle REST HTTP päringute. API serverina kasutati FastAPI[3] raamistiku. FastAPI oli piisavalt väike, aga samal ajal pakkus piisavalt

[1] https://www.postgresql.org

[2] https://postgis.net

[3] https://fastapi.tiangolo.com

funktsionaalsust nõutud eesmärgi täitmiseks. FastAPI kasuks oli veel hõlpsasti mõistetav dokumentatsioon. API server moodul tagastab pildiandmed ja kogumid GeoJSON[1] formaadis, mida on mugav *frontend* osas kasutada.

4.3.4 Veebirakenduse moodul

Veebirakenduse moodul on mõeldud lõpptulemuste visuaalseks vaatamiseks ja uurimiseks. See võimaldab saada üldise ettekujutuse ja anda vaate suuremale pildile.

Selleks, et andmeid veebirakenduses näha, pöördutakse API server mooduli poole. API server suhtleb omakorda andmebaasiga. Seejärel saadetakse leitud andmed veebirakendusele, mis kuvab punktid kaardile.

Veebirakendus on realiseeritud kasutades Reacti[2] koos TypeScriptiga[3]. Lisaks kasutatakse Material-UI[4] valmiskomponente. Kaardilahendusena kasutatakse Mapboxi[5], mis suudab mõistlikul hulgal GeoJSON punkte korraga kaardikihile kuvada. Mapboxi eelistati varasema kogemuse põhjal toetudes autori bakalaureusetööle [16], kus leiti, et Mapbox suudab efektiivsemalt kuvada suurema koguse punkte kui Google Maps[6] platvorm. Mapboxile on juurde lisatud *geocoder*[7] funktsionaalsus, mis võimaldab leida asukoha nimele vastavad koordinaadid.

Kasutajaliideses on kasutajal valida kahe GeoJSON kihi vahel mida vaadata. Esimene kiht *Photos* kuvab algselt ilma eeltöötluseta pildiandmete punkte. Teine kiht *Clusters* kuvab lõpptulemusena saadud kategoriseeritud kogumeid.

[1] https://tools.ietf.org/html/rfc7946

[2] https://reactjs.org

[3] https://www.typescriptlang.org

[4] https://material-ui.com

[5] https://www.mapbox.com

[6] https://developers.google.com/maps/documentation

[7] https://github.com/mapbox/mapbox-gl-geocoder

5 Rakendus

Järgnevalt on kirjeldatud loodud lahendust üle kõigi punktis 4.3 mainitud moodulite. See hõlmab rakendust, mis suudab CSV failist loetud pildiandmed koondada kogumiteks, kogumite pealkirjadest luua kategooriad ning kategoriseerida kogumid. Lisaks võimaldab rakendus hinnata erinevate kategoriseerimise meetodite täpsust testide alusel ja visuaalselt.

Antud töö raames loodi terve raamistik. Üheks eesmärgiks oli luua terviklikult töötav lahendus, mida on võimalik tulevikus täiustada, võimaldades uurida ja katsetada näiteks ainult ühte protsessi osa korraga. Toetudes loodud funktsionaalsusele ei tohiks olla keeruline muuta või lisada juurde uusi lahendusi.

Kogu rakenduse tööaeg kõigi Panoramio Briti saarte sisendandmetega kestis ligikaudu 54 tundi. Sellest CSV andmete ümber konverteerimine ja baasi salvestamine kestis ligikaudu viis minutit. Pildiandmete koondamine kogumiteks kestis ligikaudu 53 tundi, alternatiivse kiirema variandi kohta saab lugeda lisast 3. Kategooriate leidmine kõikide kogumite pealkirjadest kasutades spaCy teeki kestis ligikaudu 30 minutit. Kategooriate määramine kogumitele kasutades Wikipedia sissejuhatavate lõikude põhist scikit-learn TF-IDF mudelit kestis ligikaudu viis minutit. Kestvuse mõõtmiseks kasutati Intel Core i5-5200U protsessoriga ja 8GB vahemäluga arvutit.

5.1 Esialgsete pildiandmete salvestamine

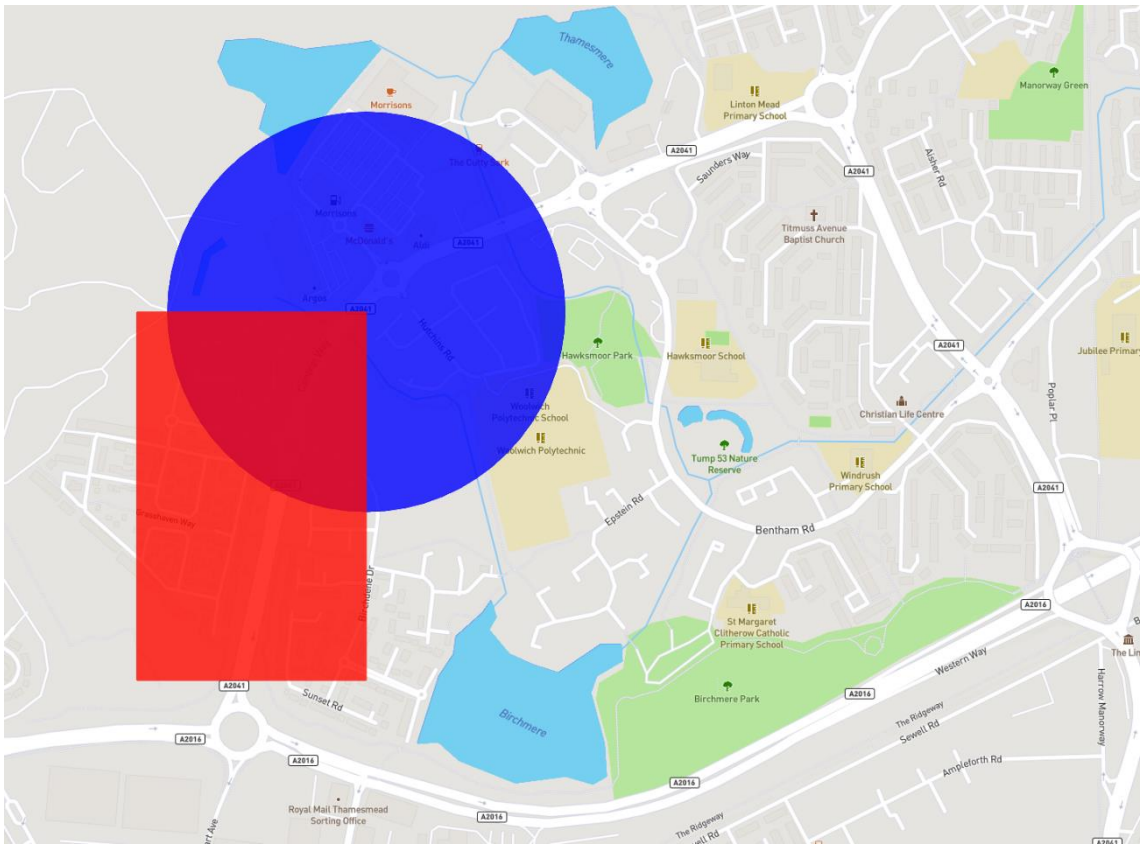
Kõige esimeseks sammuks oli pildiandmete algallikast sisse lugemine ja andmebaasi salvestamine. Antud töö skoobis tähendas see Panoramio CSV faili ridade sobivale kujule viimist ja andmebaasi salvestamist. Tulevikus on võimalik lisada juurde uusi konverteerimise viise, kui peaks olema vajadust lugeda sisse teistsugustes formaatides andmeid.

5.2 Pildiandmete koondamine

Järgmisena saab teostada sisseloetud pildiandmete koondamise kogumitesse. Ühte kogumisse koondatakse punktid, mis jäävad väljaarvutatud koordinaatide vahemikku.

Iga kogum, mis luuakse, omab esimese pildi geograafilisi koordinaate, millest kogumit hakati looma, ja sisaldab kõikide piltide pealkirju kokku aheldatult, mis jäid väljaarvutatud koondatava ala sisse. Iga uue kogumi loomine algab pildiandme punktist, kus pildiandme punkt ei ole varasemalt ühtegi kogumisse määratud. Pildiandme punkt mis on juba kogumisse määratud, enam uuesti võrdluses ei osale. See tõstab algoritmi kiirust. Koondamise algoritm töötab nii kaua, kui ühtegi punkti enam koondada ei ole.

Koordinaatide vahemik ehk ala mille sees koondamist teostatakse, on ette määratud nihkena geograafilistes kraadides. Selle põhjuseks oli tingimus, et sisendandmeid on suur kogus, seega tuleb pöörata tähelepanu algoritmi efektiivsusele. Leiti, et kiirem on arvutada nihet koordinaatides (moodustab ristkülikukujulise ala) kui raadiuse abil. Seda, kui suurt ala korraga koondatakse, saab konfiguratsiooni parameetri abil muuta. Vaikeväärtuseks seati ±0.005 kraadi, mis on mõõtmetelt umbes 345m x 550m suurune ristkülik. Võrdlus 300 meetrise raadiuse ringiga on näidatud Joonis 2 abil. Mida väiksemaks korraga koondatav ala viia, seda aeglasemaks punktide koondamine läheb. Koondatud punktid salvestatakse tabelisse, mille alusel saab hakata teostama järgnevaid samme.



Joonis 2. Koondamise ristküliku (punane) võrdlus 300m raadiuse ringiga (sinine).

5.3 Kategooriate leidmine

Kui ilma kategooriateta kogumid on olemas, saab liikuda kategooriate leidmise juurde. Sisendandmeid võib tulevikus olla vajalik laadida erinevatest allikatest ja seega leiti, et ei ole mõistlik määrata käsitsi ette, mis kategooriatesse on võimalik koondatud punkte määrata. Samas on selline võimalus olemas, kuna kategooriaid loetakse eraldi tabelist, siis piisab, kui neid soovi korral käsitsi muuta. Nii veebirakendus kui testid võtavad kasutaja poolt tehtud kategooriate muudatusi arvesse automaatselt.

Peale erinevaid katsetusi, otsustati jätta kasutajale võimalus kasutada kolme kategooriate leidmise varianti. Loodud variandid ei ole kuidagi piiratud ja kasutaja saab lisada juurde omapoolseid lahendusi. Etteantud algoritmid on rakenduse loomise ajal leitud variandid, mis peaksid olema mõistlikud ja samas täidavad näidise rolli, mille eeskujul on lihtsam luua uusi lahendusi.

Esimese ning kõige lihtsama variandi korral teostatakse kõigepealt andmete puhastus ja *stopwords* nimekirjas sõnade eemaldamine, et vältida sõnu nagu *and*, *you* jne, mida esineb tekstides väga tihti. Seejärel loetakse kokku sõnade esinemise sagedused ja järjestatakse

rohkem esinenud sõnad ettepoole. See variant annab kõige halvema tulemuse, aga ei sõltu sisendandmete keelest, kuigi mitme keele kategoriseerimise funktsionaalsus antud lõputöö skoobis ei ole, otsustati see ikkagi eeltingimuste puudumise, kiiruse ja lihtsuse tõttu alusvariandina alles jätta.

$$TF(s) = (kordi\ s\ esineb\ dokumendis) / (sõnade\ hulk\ dokumendis) \quad (1)$$

$$IDF(s) = log(dokumentide\ hulk / dokumentide\ hulk\ mis\ sisaldab\ s) \quad (2)$$

$$TFIDF(s) = TF(s) * IDF(s) \quad (3)$$

Teine ja kolmas variant kasutavad tavalise kokkulugemise asemel TF-IDF statistikale toetuvat algoritmi koos sõna algkujule viimisega. TF-IDF aitab leida olulised sõnad ning tihtiesinevad vähetähenduslikud sõnad saavad madalama statistilise väärtuse. TF-IDF (3) koosneb kahest osast TF (1) *term frequency*, kus loetakse kokku, kui tihti sõna esineb ühe dokumendi piires. Mida rohkem sõna esineb ühes dokumendis, seda kõrgem on TF. IDF (2) *inverse document frequency* korral leitakse, kui mitmes erinevas dokumendis sõna esineb. Mida vähemates dokumentides sõna esineb, seda kõrgem on IDF väärtus. TF-IDF leidmiseks kasutatakse scikit-learn[1] teegi sisseehitatud funktsionaalsust. TF-IDF on korrutis, mis üritab tasakaalustada sõna esinemissagedust dokumendis ja samal ajal tuua rohkem esile harva esinevad sõnad kõigi dokumentide piires.

Selleks, et ühest sõnast ei jääks alles mitu erinevat vormi, kasutatakse sõna algkujule viimist. Teise ja kolmanda variandi erinevus seisneb *lemma* sõna algkujude leidmises. Teises variandis kasutatakse spaCy teeki, mille abil sõna algkuju ehk *lemma* leida, ja kolmandas variandis NLTK teegi abi. SpaCy ja NLTK tulemused olid üsnagi sarnased, aga spaCy andis teatud sõnadega parema algkuju. Näiteks spaCy pakkus sõnale *looking* vormi *look*. NLTK pakkus, et *looking* võiks jääda *looking*, mis ei ole vale, aga võttes arvesse kategooriate konteksti, siis paremini sobis vorm *look*. NLTK kasuks oli kiirus, kuna tulemuste vahe ei olnud kriitiliselt suur, siis otsustati jätta alles NLTK variant, mis annab natuke halvema tulemuse, aga seda oluliselt kiiremini. SpaCy protsess kogu Panoramio andmetega kestis ligikaudu 30 minutit, NLTK-ga ligikaudu 5 minutit.

Võttes kategooriate leidmise osa lühidalt kokku, siis leitakse 100 kõige rohkem esinenud sõna, mille hulgast saab kogumitele sobivaid kategooriaid määrata. 100 kategooria piir

[1] https://scikit-learn.org/stable

on parameeter, mida on võimalik vajadusel vähendada või suurendada. Vaikeväärtuseks valiti 100 kategooriat, sest puudusid treening- ja testandmeid isetreenitud mudeli loomiseks ja testimiseks. Et oleks võimalik mõistliku ajaga treening- ja testkogumid märgendada, siis ei saanud viia kategooriate valikut liiga suureks.

5.4 Kogumite kategoriseerimine

Kui ilma kategooriateta kogumid ja 100 enim esinenud sõna ehk kategooriat on olemas, saab leida, millised kategooriad 100-st on igale kogumile kõige sobivamad. Igale kogumile määratakse maksimaalselt kolm kategooriat. Kolme kategooria piiri saab kasutaja parameetri abil muuta. Otsustati seada vaikeväärtusena maksimaalselt kolm kategooriat, et testimine oleks võimalikult võrdne erinevate algoritmide vahel.

Punktide kategoriseerimine toimub kogumite pealkirjade alusel. Piltide pealkirjad on tavaliselt lühikesed ja pole eriti sisukad, seega ei pruugi need alati anda edasi piisavalt konteksti. Sellepärast otsustati kategoriseerimine teostada hetkel, kus pildiandmed on kogumiteks koondatud ja kus iga kogumi pealkiri koosneb kogumis sisalduvate piltide pealkirjadest üksteise järel ahelasse panduna [17], [18], [19].

Kategoriseerimiseks kasutati ühte ilma mudelita algoritmi, masinõppe põhiselt nelja eeltreenitud mudelit, kahte TF-IDF põhist mudelit ja kahte isetreenitud mudelit. Eeltreenitud mudelite suurem hulk võrreldes isetreenitutega on tingitud sellest, et sobilikke treening- ja testandmeid ei leitud, mille abil saaks isetreenitud mudeleid treenida ja testida. Otsustati ikkagi katsetuse eesmärgil luua kaks isetreenitud mudelit, mida treeniti autori poolt käsitsi kategoriseeritud 100 treeningkogumi pealt.

Eeltreenitud mudeliteks olid TensorFlow *Universal Sentence Encoder*[1] [20], Stanford GloVe[2] [21], Facebook fastText[3] [22], [23], Explosion AI spaCy[4] vektormudelid. TF-IDF

[1] https://tfhub.dev/google/universal-sentence-encoder/4

[2] https://nlp.stanford.edu/projects/glove

[3] https://fasttext.cc/docs/en/crawl-vectors.html

[4] https://spacy.io/models/en#en_core_web_md

statistikale põhinevad mudelitest esimene loodi ainult kogumi pealkirjade alusel ja teine Wikipedia sissejuhatavate tekstide alusel[1] [24].

$$cosine\ similarity = \cos\theta = \frac{A * B}{\|A\| * \|B\|} \quad (4)$$

GloVe, spaCy, fastText ja *Universal Sentence Encoder* mudelite kategooriate leidmise juures kasutatakse *cosine similarity* (4) meetrikat. See võimaldab arvutada vektorite suunalist erinevust punktide vahel vektorruumis. Üheks vektoriks on pealkirja vektor ja teiseks kategooria vektor. Mida väiksem on nurk kahe vektori vahel, seda suurem on kontekstiline sarnasus. Mida suurem on nurk kahe vektori vahel, seda väiksem on *cosine similarity* väärtus ja seda väiksem on kontekstiline sarnasus tekstide vahel.

Kuigi scikit-learn teegis on *cosine similarity* leidmise funktsioon olemas ja toimib vastavalt valemile, siis katsetuse tulemusena leiti, et luues oma funktsioon toetudes numpy teegile oli see oluliselt kiirem. Piisav vahe esines juba väikeste koguste juures. Kategoriseeriti 1000 kogumit scikit-learn funktsiooniga ja numpy teegi abil loodud funktsiooniga. Tulemused olid samad, aga scikit-learn funktsioon kestis 35 sekundit ja numpy teegi abil loodud variant 12 sekundit. Tasub märkida, et 1000 kogumi kategoriseerimine on väike hulk andmeid ja mida rohkem kogumeid kategoriseerida, seda suuremaks ajalised vahed nihkuvad. Kahe funktsiooni ajalist mõõtmist teostati *Universal sentence encoder* mudelit kasutades.

5.4.1 Sõna sageduse põhine algoritm

Sõna sageduse põhine algoritm on üks lihtsamatest, väiksema ressursikasutusega ja kiiremaid variante. Sisuliselt puhastatakse, tükeldatakse ja sorteeritakse kogumi pealkirjadest saadud sõnad esinemissageduste alusel. Enim esinenud sõnad kogumi pealkirjas sorteeritakse ettepoole. Seejärel leitakse kogumi pealkirjas esinenud sõnadest kolm esimest vastet 100-st kategooria sõnast.

5.4.2 Sisendandmetest loodud scikit-learn TF-IDF mudel

Mudel on loodud kasutades scikit-learn raamistikku. Tegu on TF-IDF statistikal põhineva mudeliga, mille vektorid on loodud kogumite kõikide pealkirjade alusel. Kasutatakse scikit-learn *CountVectorizer*'i ja *TfidfTransformer*'it. Selleks, et leida vasted, leitakse TF-

[1] https://thijs.ai/Wikipedia-Summary-Dataset

IDF mudelist 10 kõige kõrgema TF-IDF skooriga sõna, ühe kogumi pealkirja piires, mida üritatakse kategoriseerida. Need on järjestatud suurema skooriga sõnad eespool. Seejärel leitakse maksimaalselt kolm esimest vastet 100-st kategooria sõnast.

5.4.3 Wikipedia sissejuhatavate lõikude põhine scikit-learn TF-IDF mudel

Mudel on sarnane punktis 5.4.2 kirjeldatud variandile. Erinevus seisneb andmetes, mille alusel on mudel loodud. Kui punktis 5.4.2 moodustati vektorid kogumite pealkirjadest, siis selles mudelis kasutatakse *Wikipedia-Summary-Dataset'*i ehk Wikipedia sissejuhatavaid tekste, mis on varasemalt puhastatud. Leiti, et alates 100 000 sissejuhatavast tekstist mudeli tulemus paremaks ei läinud, seega määrati see sisseloetavate lõikude piirväärtuseks. Wikipedia sissejuhatavate tekstide koguse piiramine võimaldab muuta mudel kiiremaks ja samas kasutab vähem vahemälu. Soovi korral saab kasutaja sissejuhatavate tekstide koguse piirangu väärtust parameetri abil muuta.

5.4.4 Isetreenitud scikit-learn TF-IDF mudelid

Mudelid põhinevad scikit-learn raamistiku TF-IDF mudelil. Võimalik on valida kahe erineva klassifikaatoriga mudeli vahel. Esimeseks valikuks on *multiclass OneVsRestClassifier* koos *DecisionTreeClassifier'*iga ja teiseks valikuks *neuralnet MLPClassifier*. Katsetati veel teisi scikit-learn teegi poolt pakutavaid mitme kategooriaga klassifikaatoreid, aga loodud testide alusel andis proovitud variantidest ühe parema tulemuse *DecisionTreeClassifier*. *MLPClassifier* ei andnud head tulemust, aga lisati võrdlusmomendi tekitamiseks scikit-learn *neuralnet* mudeleid esindades.

DecisionTreeClassifier ehk otsustuspuu korral arvutatakse ja liigutakse kõige parema tulemuse suunas, liikudes mööda otsustusharusid. Otsustuspuu mitme kategooriaga probleemi korral luuakse iga *label'*i ehk kategooria kohta eraldi puu ja leitakse millise kategooria puu annab kõige kõrgema tulemuse.

MLPClassifier (*Multi-layer Perceptron classifier)* treenitakse iteratiivselt niikaua kui soovitud kogus iteratsioone on tehtud. Iga iteratsiooni käigus arvutatakse vea funktsiooni BFGS (*Broyden–Fletcher–Goldfarb–Shanno algorithm*) abil välja, kas iteratsiooniga treeniti mudel paremaks või halvemaks, sellele vastavalt muudetakse iga iteratsiooni käigus mudelisisesed kaale.

Treeningandmeteks olid käsitsi märgendatud 100 treeningandmetena kasutatavat kogumit. Treeningandmete kogus oli väike, aga sobivate testandmete puudumisel loodi see kontseptsiooni tõestuse eesmärgil ning võrdluse saamiseks teiste mudelitega. Parema mudeli treenimiseks on tõenäoliselt vaja oluliselt rohkem andmeid, kuna antud töö põhieesmärgiks ei olnud andmete märgendamine, siis piirduti 100 treeningkogumi ja 100 testkogumi märgendamisega.

5.4.5 TensorFlow *Universal Sentence Encoder* eeltreenitud mudel

Mudelina kasutatakse TensorFlow eeltreenitud *Universal Sentence Encoder* versioon 4 mudelit. Kõik kogumite pealkirjad kodeeritakse *Universal Sentence Encoder* abil 512 dimensioonilisteks vektoriteks. Seejärel kodeeritakse 100 kõige rohkem esinenud kategooria sõna samuti *Universal Sentence Encoder* abil vektoriteks. Peale seda leitakse maksimaalselt kolm kõige sarnasemat kategooriat. Sarnasuse leidmiseks kasutatakse *cosine similarity* funktsiooni. *Universal Sentence Encoder* üks kasutusaladest on tekstide sarnasuse võrdlemine.

5.4.6 Explosion AI spaCy eeltreenitud mudel

Eeltreenitud mudelina kasutatakse spaCy keskmise suurusega inglisekeelset veebitekstide mudelit, mis oli kiiruse osas veel piisavalt kiire kohalikus arvutis kasutamiseks. SpaCy inglisekeelsete veebitekstide mudelitest oli valida kolme eri suurusega mudeli vahel, väike, keskmine ja suur. Väikses mudelis puudusid vektorid, mille abil sõnu võrrelda, ning suur mudel oli liiga aeglane ja mahukas kohalikus arvutis käivitamiseks.

SpaCy eeltreenitud mudel põhineb Common Crawl ja OntoNotes 5 allikatel, vektorid on 300 dimensioonilised. Leidmaks kõige sarnasemaid märksõnu kogumite pealkirjadele kasutati spaCysse sisseehitatud *cosine similarity* funktsiooni. See leiab sõna sarnasuse kogumi pealkirjale. Kuna kategooria sõna koosneb ühest sõnast ja kogumi pealkiri on tavaliselt rohkem kui üks sõna, siis sel juhul toimib spaCy sisseehitatud võrdluse funktsioon nii, et võetakse pealkirja sõnade kõikide sõnade vektorite keskmine.

5.4.7 Facebook fastText eeltreenitud mudel

Mudelina kasutatakse Facebooki fastText eeltreenitud ingliskeelset mudelit, mis on eeltreenitud Common Crawl ja Wikipedia andmete pealt, vektorid on 300

dimensioonilised. Kategooria tabeli sõnade ja kogumi pealkirjade sarnasuse leidmiseks kasutatakse *cosine similarity* funktsiooni. Kogumi igale sõnale on leitud vastav vektor ja liidetud kõik sõnade vektorid kokku ning jagatud sõnade arvuga ehk on võetud sõnade vektorite keskmine. Kategooria tabeli sõnadele on samuti määratud vektorid fastText mudelist, et oleks võimalik võrrelda kogumite pealkirju kategooriatega.

5.4.8 Stanford GloVe eeltreenitud mudel

GloVe on eeltreenitud mudelitest kõige vanem. Eeltreenitud mudeli allikatena on kasutatud Wikipedia 2014 ja Gigaword 5, kuue miljardi sõnaga. Mudel sisaldab 400 000 erinevat sõna 50 dimensiooniliste vektoritega. Kogumi igale sõnale määratakse oma vastav vektor ja kui tegu on mitmesõnalise fraasiga, siis leitakse sõnade vektorite keskmine. Samuti määratakse vektorid kategooriatele, et kogumi pealkirjade vektoreid oleks võimalik kategooriatega võrrelda.

Lisaks Wikipedia 2014 ja Gigaword 5 kuue miljardi sõnaga treenitud vektormudelile katsetati Common Crawl põhjal treenitud 42 miljardi sõnaga mudelit. Suurem mudel sisaldas 1,9 miljonit erinevat sõna 300 dimensiooniliste vektoritega, aga see paremaid tulemusi ei andnud. Põhjuseks võib olla Common Crawl mürarikkam andmete kvaliteet võrreldes Wikipedia tekstidega. Kuna tulemused ei olnud paremad ja ressursi kasutus oli suurem, otsustati piirduda Wikipedia 2014 ja Gigaword 5 kuue miljardi sõnaga mudeliga mis oli väiksema ressursikasutusega.

5.5 Tulemuste hindamise viisid

Tulemuste hindamiseks loodi kaks viisi. Esimene viis annab konkreetse ning kiire hinnangu väiksele hulgale testidele põhinedes. Teine variant on üldisem ja mille tulemusi konkreetselt ei saa hinnata, aga võimaldab anda visuaalse hinnangu saadud tulemustele, kus töödeldud punktide hulk saab olla oluliselt suurem.

5.5.1 *F-score* testid

Tulemuste valideerimiseks loodi testide süsteem. Kuna andmeid on palju, siis ei ole võimalik kiirelt kõigi erinevate mudelite ja algoritmide headust kogu andmehulga pealt hinnata. Testid annavad konkreetse ja kiire indikatsiooni milline kategoriseerimise meetod annab täpsema ja kiirema tulemuse. Testidena kasutatakse 100 käsitsi valitud ja märgendatud testkogumit. Tulemuse hindamiseks kasutatakse *f-score* väärtust.

$$Fscore = 2 * (precision * recall) / (precision + recall) \quad (5)$$

F-score arvutamisel kasutatakse scikit-learn teegi *f-score* (5) funktsionaalsust. Nagu valemist näha, koosneb *f-score* täpsusest (*precision*) ja saagisest (*recall)*. Täpsus näitab, kui palju oli leitud tõestest tulemustest tegelikult tõesed. Saagis näitab, kui palju oli kõigist tõestest tulemustest tegelikult leitud.

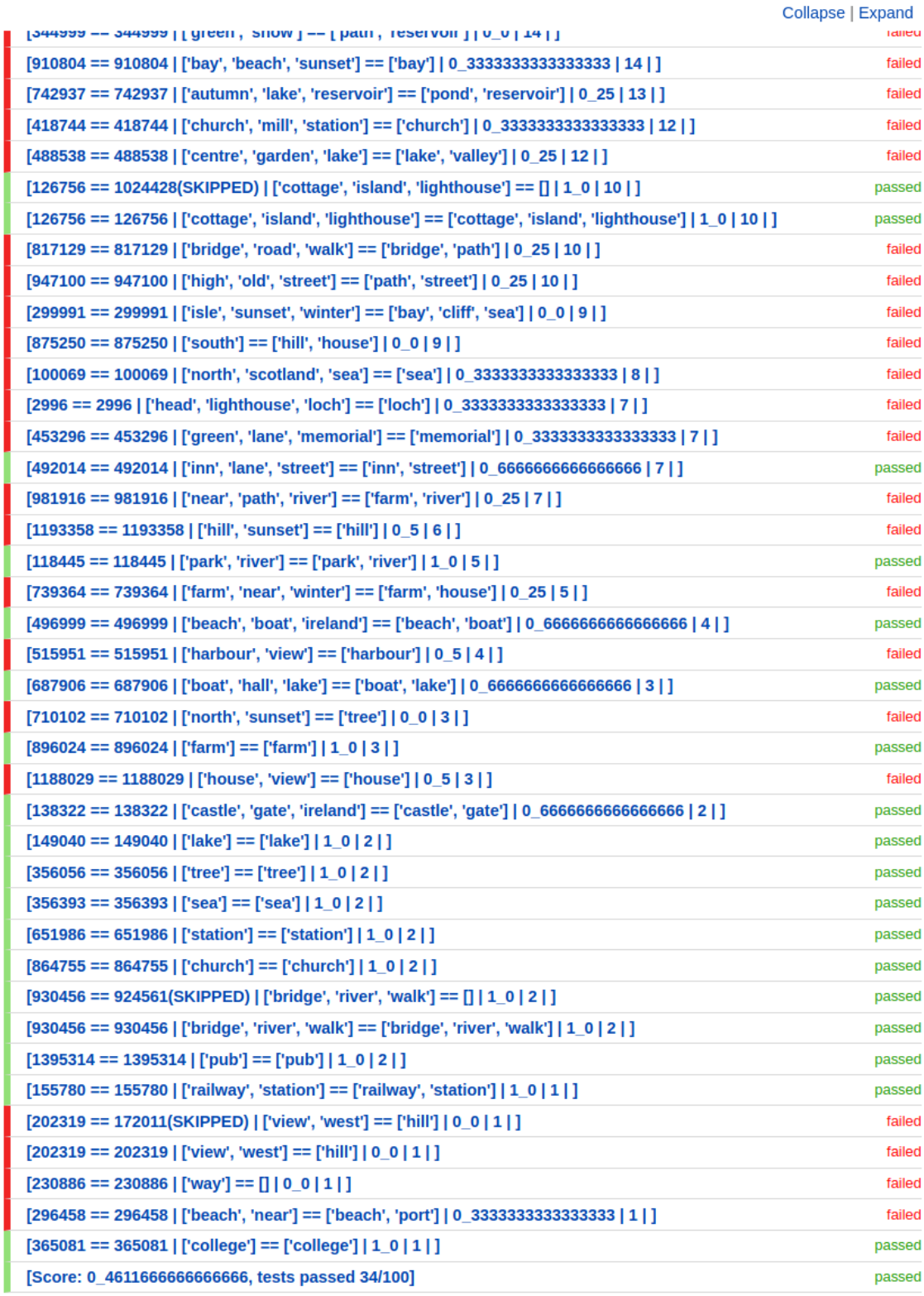
Mitme kategooria klassifitseerimise juures kasutatakse scikit-learn teegi *f-score* funktsiooni koos sisseehitatud *micro-average* keskmise arvutamise parameetriga. *Micro-average* loeb kokku kõik tõeselt positiivsed (*true positive*), valepositiivsed (*false positive*) ja valenegatiivsed (*false negative*) tulemused[1] sõltumatult, ühe testi piires. Lisaks on vaja mitme kategooriaga klassifitseerimise juures määrata, mida teha, kui *f-score* arvutamisel on ühes võrreldavas järjendis vähem elemente kui teises. See saab juhtuda näiteks siis, kui algoritm leiab ühe sobiva kategooria, aga kasutaja poolt on määratud kolm kategooriat. Sel juhul määratakse puuduvatele kohtadele järjendis tühi sõne. See tähendab, et võrdluse hetkel võrreldakse ühe järjendi kindlat elementi teise järjendi tühja sõnega, mis annab võrreldud elementidega madala tulemuse, mis oligi eesmärgiks.

Järgnevalt on Joonis 3 abil kujutatud ühe mudeli 100 alamtesti tulemuste unittest raamistiku genereeritud vaadet. Joonisel on kuvatud mittetäielik vaade 100 alamtestist. Genereeritud testimise tulemuste vaate eripärade tõttu on numbrite vahel olevad komad asendatud alakriipsudega. Testida on võimalik kõiki mudeleid korraga või iga mudelit eraldi.

Igal real on alamtesti järgmised parameetrid: kogumi mudeli poolt pakutud id == tegelik id, mudeli pool pakutud kategooriad == tegelikud kategooriad, *f-score* tulemus alamtesti piires, kogumi populaarsus ehk mitu pildiandme punkti kogumisse lisati koondamise sammus. Viimane rida näitab *f-score* tulemust ühe mudeli piires ehk 100 alamtesti peale kokku. *Test passed* näitab, mitu alamtesti sai *f-score* tulemuse üle 0.5. Kogumi id võrdlus on toodud välja, et oleks võimalik saada aru, millistele kogumitele mudel ühtegi kategooriat ei seadnud. Read, kus id taha on märgitud *SKIPPED*, tähendab, et mudel ühtegi kategooriat kogumile ei määranud. Näiteks id 924561 juures on näha, et alamtest sai skoori 1,0, sest oodatud tulemus pidigi olema tühi. Samas id 172011 juures oli skoor

[1] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

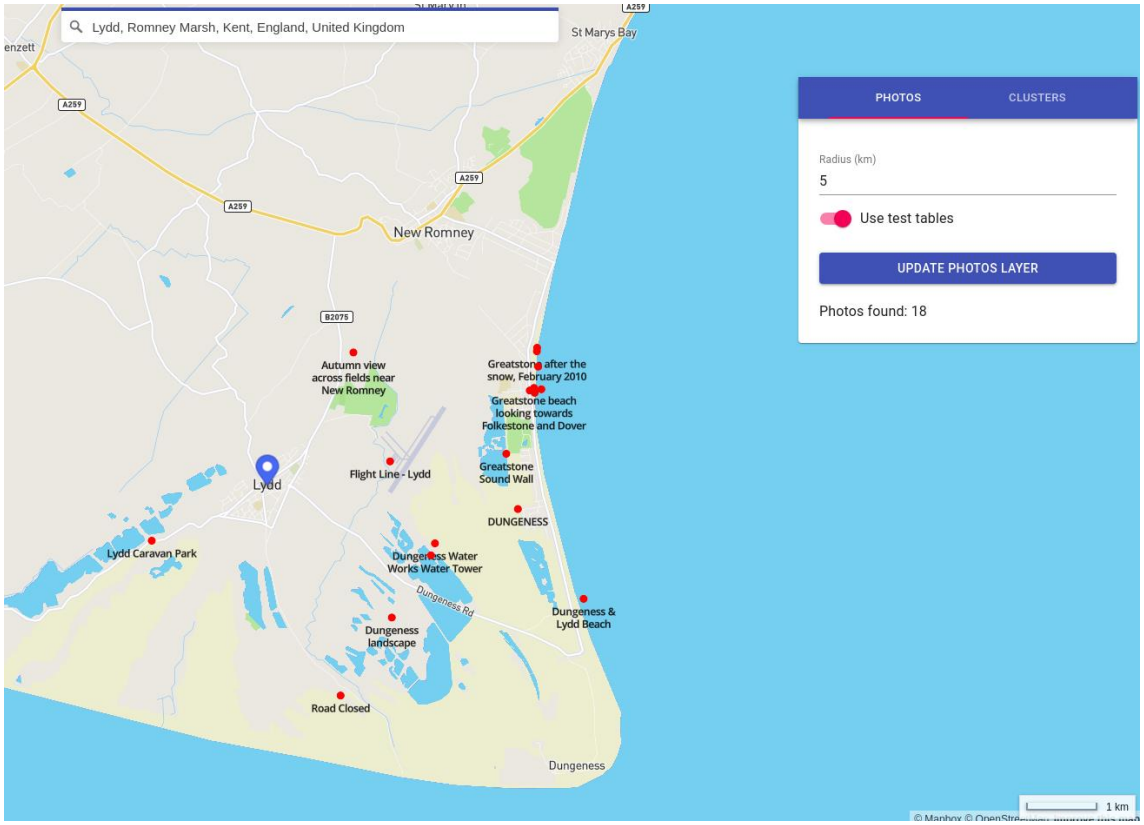
0,0, sest oodati tulemuseks *hill*, aga mudel ühtegi kategooriat 172011 id-ga kogumile ei määranud.



Joonis 3. Lõik 100 alamtesti tulemuste genereeritud vaatest mudeli kohta.

5.5.2 Veebirakendus

Lisaks testidele loodi veebirakendus, mis annab ülevaate saadud tulemustest visuaalselt. See võimaldab uurida tulemusi ja on testidele toetavaks mehhanismiks. Järgnevalt on Joonis 4 abil näidatud, kuidas veebirakenduse liides välja näeb. Võimalik on vaadata salvestatud ilma eeltöötluseta pildiandmete punkte *photos* saki alt ja töötluse täielikult läbinud ehk koondatud ja kategoriseeritud kogumite punkte *clusters* saki alt. Nii *photos* kui *clusters* tulemusi on võimalik vaadata põhitabelitest või testtabelitest muutes *Use test tables* lüliti olekut. Kasutajaliidese abil on võimalik otsida asukohta asukoha nime järgi. Lisaks on võimalik määrata filtreerimise raadius otsitud punktist. Täielikult töödeldud kogumite korral on võimalik lisaks raadiusele täiendavalt filtreerida populaarsuse ja kategooriate järgi. Rohkem näiteid kasutajaliidesest on toodud punktis 6.2.



Joonis 4. Veebirakenduse vaade *photos* sakil testtabeli andmetega.

6 Tulemused

Järgnevalt hinnatakse ja võrreldakse loodud algoritmide ning mudelite tulemusi. Hindamiseks kasutatakse 100 testkogumit. Lisaks 100 testkogumile märgendati isetreenitud mudelite treenimiseks veel 100 treeningkogumit. Treeningkogumid on eraldiseisvad 100 kogumit, mis on mõeldud mudelite treenimiseks. Treening- ja testkogumite andmed on erinevad ja ei kattu.

Test -ja treeningkogumiteks üritati valida võimalikult erinevaid asukohti, populaarsust ja teksti sisu, et katta võimalikult palju tingimusi ja mõistlikke kategooriaid mis valikuteks olid. Autor seadis märgendades endale samad tingimused nagu oli seatud algoritmidele ja mudelitele ehk maksimaalselt oli lubatud ühele kogumile märkida kolm kategooriat. Kui kogumi pealkirja sisu ei sobinud ühegi 100 kategooriaga, siis jäeti kogum ilma kategooriateta.

Sisendandmeteks oli 1,41 miljonit Panoramio pildiandme CSV rida. Sellest jäi peale koondamist kogumitena alles umbes 136 000 kategooriateta kogumit. Peale kategooriate leidmist, kasutades spaCy teeki, ja kategoriseerimist, kasutades Wikipedia sissejuhatavate lõikude põhist scikit-learn TF-IDF mudelit, jäi alles umbes 94 000 ühe kuni kolme kategooriaga kogumit. Kogumid, millele mudel ühtegi vastet 100-st valitavast kategooriast ei leidnud, lõpptulemustesse alles ei jäetud.

6.1 *F-score* tulemused

Tulemustes on välja toodud üheksa erinevat varianti kogumite kategoriseerimiseks. Joonis 5 abil on näha kõikide kirjeldatud variantide tulemusi 100 testkogumit kategoriseerides.

Alustades tulemuste analüüsi, halvematest paremate tulemuste suunas, siis on näha, et kõige halvema tulemuse andis isetreenitud-TF-IDF-neural, mis kasutas scikit-learn teegi närvivõrgu klassifikaatorit. Halva tulemuse põhjuseks võib olla vähene treeningandmete hulk. Järgmiseks on GloVe ja spaCy, mille tulemused on üsna lähestikku.

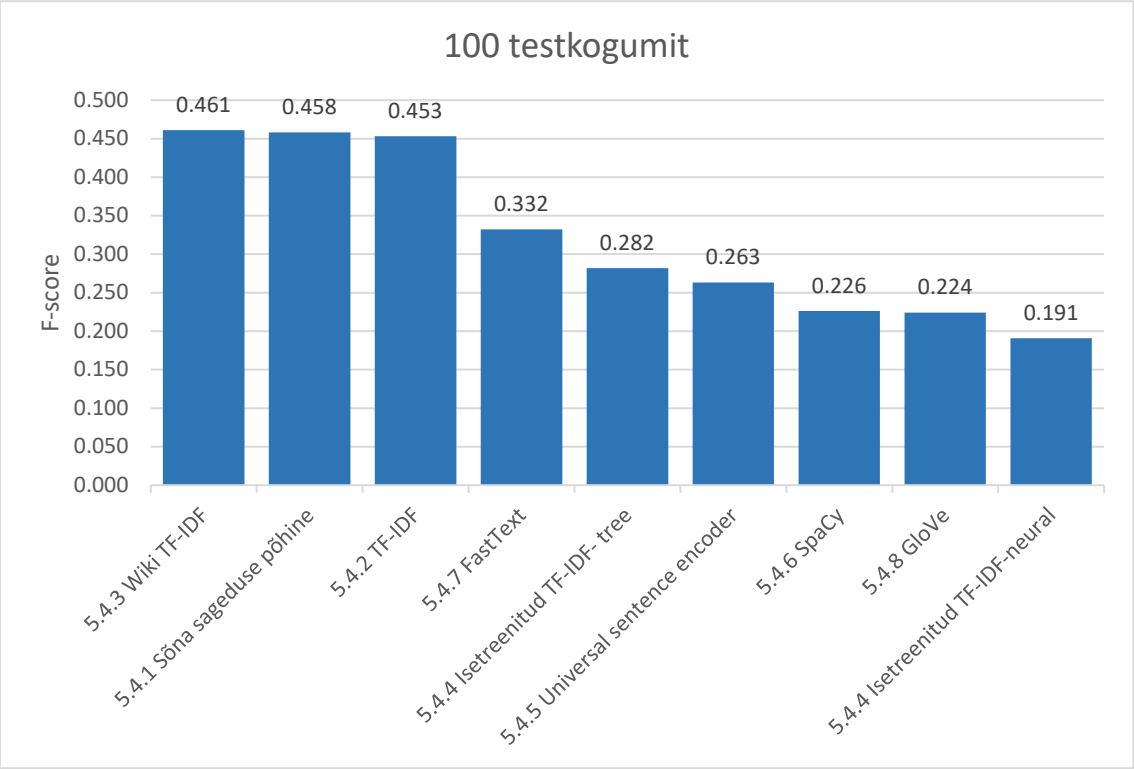
Universal Sentence Encoder tulemus ei ole kiita, kuigi see on optimeeritud lausete võrdluseks. Alguses oli hüpotees, et antud mudel võiks olla üks parimatest variantidest, aga nagu näha, siis on see pigem halvemate tulemuste seas. Põhjuseks võib olla pildi pealkirjade liiga vähese kontekstiga tekstid, millest ei ole võimalik luua piisavalt selgelt eristatava kategooriatega vektoreid.

Peale *Universal Sentence Encoder*'it on järgmine isetreenitud mudel, isetreenitud TF-IDF-tree, mis treeniti treeningkogumite pealt kasutades scikit-learn teegi otsustuspuu klassifikaatorit. Kuigi tulemus on kõikide tulemuste seas keskel, siis antud mudelil võib olla kõige rohkem potentsiaali. Saadi tulemus, mis pole küll parim, aga pole ka kõige halvem, kasutades selleks ainult 100 treeningkogumi andmeid. Kui tulevikus on võimalik lisada juurde rohkem treeningandmeid, siis on tõenäosus, et antud mudel toimib oluliselt paremini.

FastText on testide alusel kõige parem eeltreenitud mudel mida prooviti. Kuigi tulemus üleüldiselt ei ole olnud parim, andis see kõigist eeltreenitud variantides kõige parema tulemuse.

Esikolmikusse jäid kaks TF-IDF statistikale põhinevat mudelit ja ilma mudelita sõna sageduse põhine algoritm. Nagu näha, siis ei saa alahinnata ilma mudelita meetodi ehk sõna sageduse põhist algoritmi mudelitega variantide suhtes. Üldpilti vaadates olid ka kõige paremad tulemused kesised. Parima tulemuse andis TF-IDF mis loodi ingliskeelsete Wikipedia sissejuhatavate tekstilõikude abil. Wikipedia tekstide mudel võrreldes ainult kogumite pealkirjade pealt loodud TF-IDF mudeliga, mis jäi kolmandaks, parandas *f-score* tulemust 0.008 võrra.

Wiki-TF-IDF scikit-learn mudelit katsetati mitmel viisil. Prooviti mudelit luues muuta lõikude sissejuhatavate sisseloetud koguseid. Leiti, et peale 100 000 sissejuhatavat lõiku tulemused oluliselt ei paranenud. Veel prooviti mudelit luua nii, et vektorite loomiseks kasutati ainult sissejuhatavaid lõike, mis sisaldasid sõnu nagu *uk, United kingdom*, *British isles*, *Britain*, aga taoline kitsendamine paremat tulemust ei andnud.



Joonis 5. 100 testkogumi tulemused.

Võttes kokku saadud *f-score* tulemused, siis on näha, et skoorid ei ole just kõrged. Suure tõenäosusega on tulemuste skoorid madalad, sest pole piisavalt sobival kujul treeningandmeid, mille pealt oleks võimalik treenida korralik mudel. Otsustati katsetada eeltreenitud mudeleid. Töö käigus leiti, et need ei anna head tulemust. Siin oli tõenäoliseks põhjuseks probleem, et piltide pealkirjad on liiga üldised ja neid on eeltreenitud mudelite alusel keeruline täpselt kategoriseerida, sest need pole treenitud spetsiifiliselt Panoramio pildiandmete pealkirjade alusel. Prooviti ka variante, mis ei nõudnud spetsiifilisi treeningandmeid ja eeltreenitud mudeleid, nagu sõna sageduse põhine algoritm ja TF-IDF statistikal põhinevad mudelid. Nagu tulemustest näha, siis need olid katsetatud variantides kõige paremad.

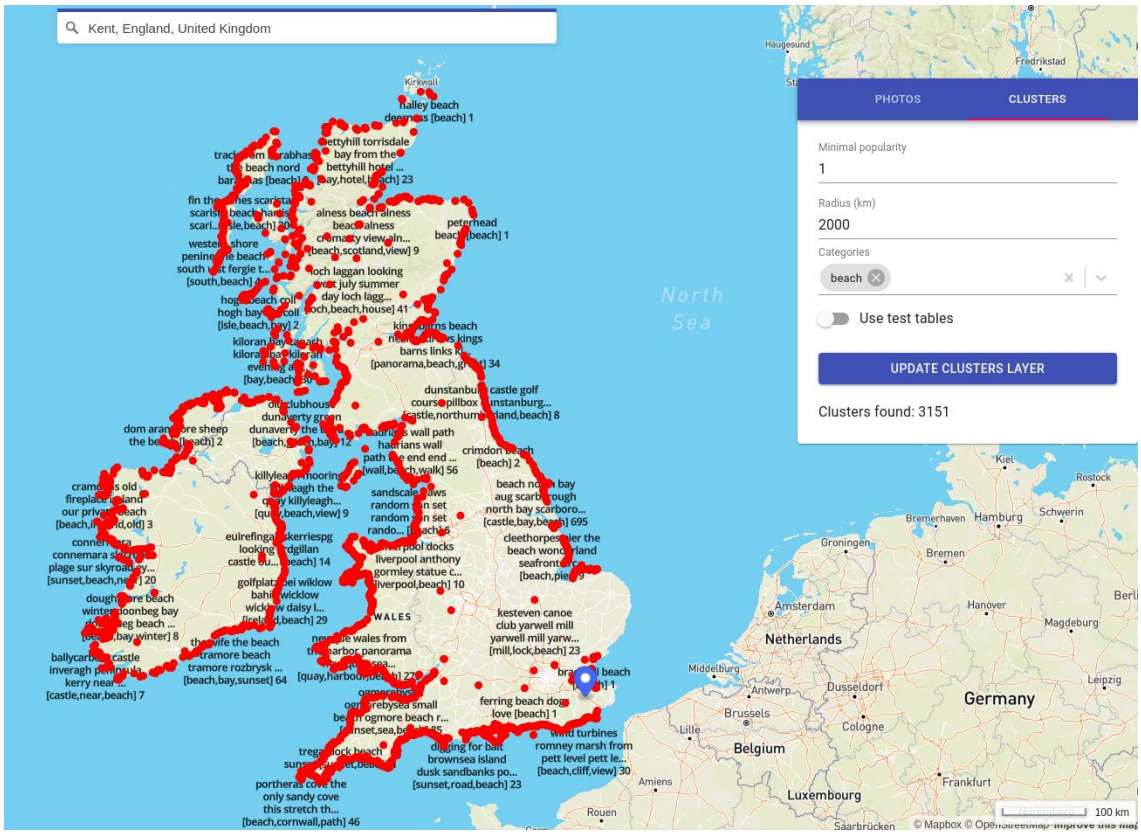
Ressursikasutuse osas kasutas testide ajal kõige rohkem vahemälu fastText. Keskmise kasutuse alla võib liigitada GloVe, spaCy ja *Universal Sentence Encoder*'i. Ülejäänud variandid olid pigem väikse mälukasutusega.

Kiiruse osas olid kõige kiiremad sõna sageduse põhine ja TF-IDF variandid. Eeltreenitud mudelite korral võttis alguses aega mudelite vahemällu lugemine ja üleüldiselt olid eeltreenitud mudelid võrdlemises aeglasemad sõna sageduse põhise ja TF-IDF variantidest.

6.2 Veebirakenduse kaudu nähtav tulemus

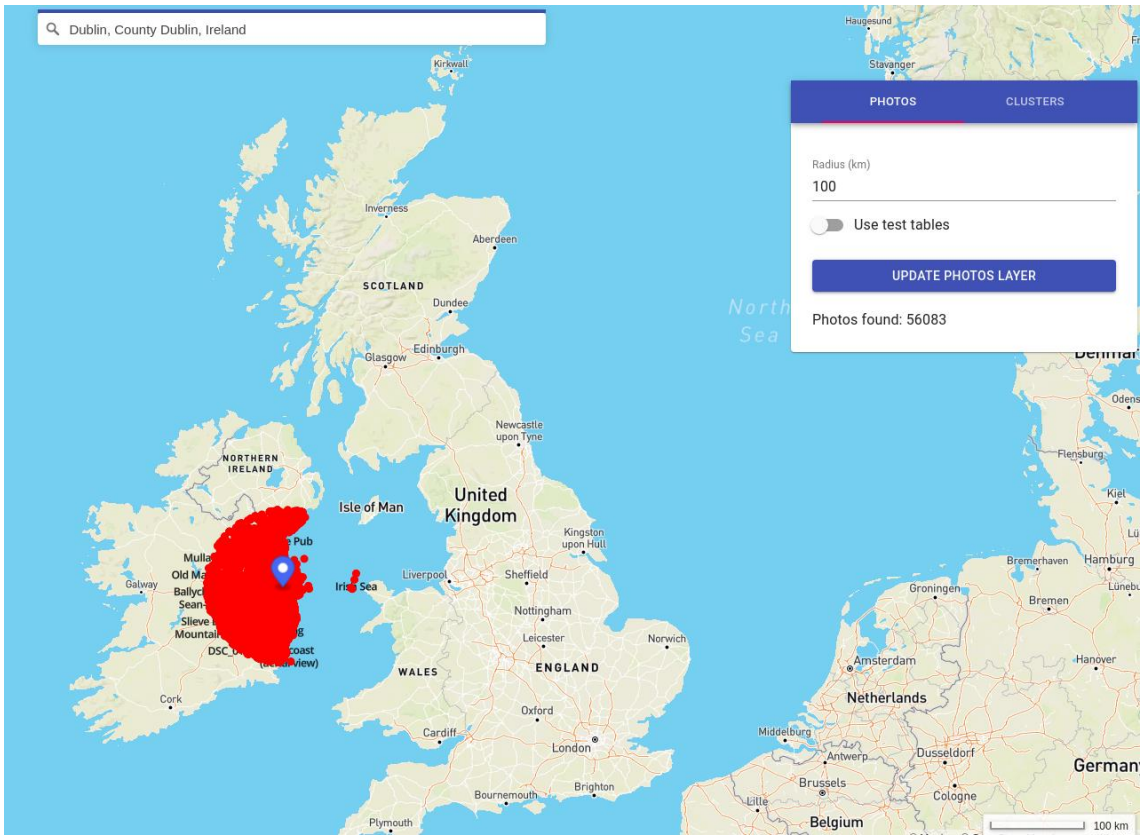
Järgnevalt on Joonis 6 abil toodud näide veebirakenduse vaatest, kus kategoriseerimine on teostatud Wiki TF-IDF mudeli abil. Kasutajaliidese abil on valitud Mapbox *geocoder*'it kasutades asukohaks Kent, England, United Kingdom. Veel on valitud, et kuvataks ainult kogumeid, mille üheks kategooriaks on *beach*, ja seda Kentist 2000 kilomeetri raadiuses, et näidata kõiki leitud randu kogu Briti saarte ulatuses. Lisaks on võimalik piirata kogumeid *minimal popularity* väärtuse abil, et kuvataks ainult määratud väärtusega võrdseid või suurema pildiandmete kogusega kogumeid. Iga kogumi kohta on näha pealkirja, kategooriaid ja populaarsust. Kogumi pealkirjast kuvatakse maksimaalselt 50 esimest sümbolit, et nähtav veebirakenduse vaade ei oleks kaetud tervenisti kogumi pealkirjade tekstidega, sest need võivad olla väga pikad, kui sisaldavad paljude pildiandmete pealkirju. Populaarsus näitab, mitu pildiandme punkti kogumisse töötluse käigus oli lisatud.

Nagu näha, siis teatud muster on välja kujunenud, suurem osa kogumeid, mille vähemalt üks kategooria on *beach* ehk rand, on mere ääres. Mõningaid kogumid asuvad sisemaal, kus võib olla tegu järvede, jõgede või muude siseveekogude randadega. Inglismaa idaranniku äär on tühi, sest sealt puuduvad erandina Panoramio algsed pildiandmed. Parema ettekujutuse puuduvatest punktidest saab vaadates Lisa 4, kus on kuvatud kõik kogumid.



Joonis 6. Vaade kogumitest *beach* kategooriaga terve Briti saarte ulatuses.

Joonis 7 kujutab eeltöötluseta salvestatud pildiandmeid Dublinist 100 kilomeetri raadiuses. Töötlemata pildiandmete koguhulk on oluliselt suurem kui lõpptulemusena kategoriseeritud kogumite arv. Iga punkt kujutab endast ühte pilti mille kohta kuvatakse pildi pealkirja.



Joonis 7. Vaade algsetest salvestatud pildiandmetest Dublinist 100km raadiuses.

6.3 Andmebaasi salvestatud lõplik tulemus

Järgnevalt on toodud Tabel 2 abil lühike lõik lõpptulemusena saadud andmetest. Tulemuste saamiseks on läbitud Panoramio CSV konverteerimine sobivateks objektideks, objektide koondamine kogumiteks, spaCy teegi abil kategooriate leidmine ja Wikipedia sissejuhatavate lõikude põhise scikit-learn TF-IDF mudeli abil kategoriseerimine. Näidisena on valitud ainult väiksemad kogumid, et need tabelisse ära mahutada. Tabelis on näha kogumid pealkirja, määratud kategooriate, asukoha ja populaarsuse infoga.

Tabel 2. Lõik lõppandmetest andmebaasis.

id	pealkiri	kategooriad	laiuskraad	pikkuskraad	populaarsus
484	sheiling diobadail isle lewis	isle	58.410716	-6.191815	1
493	view from gladstone house	view, house	58.480773	-6.228862	1
506	lighthouse	lighthouse	58.502664	-6.272850	1
549	old post office	old	58.479236	-6.234816	1
592	stacks near cross isle lewis	isle, near	58.483286	-6.299973	1
645	rather large free presbyterian church scotland building cross the ness area nort	scotland, church	58.468297	-6.294651	1
677	mangurstadh stormy mangurstadh cliff spray mangurstadh	cliff	58.165768	-7.092619	3
1222	whats happen cow punk gallanhead hotel aird uig	hotel	58.226585	-7.023354	2
1468	looking back towards uigen isle lewis	isle	58.198711	-6.947308	1

7 Kokkuvõte

Töö eesmärgiks oli luua raamistik koos mudelitega, mis võimaldab kiiresti võrrelda erinevate piltide kategoriseerimiste mudelite või algoritmide headust, pakkudes samal ajal lihtsat laiendamist uute mudelite ja testide lisamise näol. Raamistik on vajalik Sightsmap rakenduse tarbeks, et jätkata turismiobjektide rakenduse arendust. Töös seati kolm alameesmärki.

Esimeseks alameesmärgiks oli kõigepealt teha selgeks vajadused ja luua vastav raamistik. Tuli luua ühtse stiili alusel šabloonrakendus, mis lihtsustaks erinevate sammude lisamist või muutmist tulevikus. Rakendus koosneb eraldiseisvatest moodulitest, mida on vajadusel lihtne välja vahetada ja võimalik käivitada eraldi konteineritena serveris.

Teiseks eesmärgiks oli uurida ning lisada loodud raamistikule erinevaid mudeleid, tagamaks, et tehtud lahendus toimib nii nagu soovitud. Autor kasutas raamistikku ise samamoodi nagu saavad hakata seda kasutama tulevikus need, kes Sightsmap kategoriseerimise teemaga tegelevad. See andis võimaluse näha, kas loodud lahenduses on kitsaskohti, mida koheselt paremaks teha. Lisaväärtusena raamistiku katsetamisega, ja omapoolsete mudelite ning algoritmide lisamisega, loodi mustrid tuleviku kasutajatele, mis võimaldab neil näidete alusel kiiremini kohandada oma ideed ja mudelid raamistikule vastavaks. Mudelite ja algoritmide väljatöötamine oli üks suuremaid osasid antud lõputöös, sest katsetamist oli palju. Kõige keerulisem oli leida töötavaid lahendusi, samal ajal võttes arvesse piiranguid, et kõik protsessi sammud peavad olema piisavalt kiired ja kategoriseerimise osas puudusid testandmed mudelite treenimiseks.

Kolmandaks ja viimaseks eesmärgiks oli luua testimise süsteem, mis annaks kiire esialgse tagasiside proovitud meetodi headusest *f-score* tulemusena ja veebirakenduse kaudu visuaalselt. Kiire tagasiside võimaldab anda kiire esialgse hinnangu, milliste mudelitega tasub tööd jätkata ning rohkem vaeva näha ja mida tasub välistada või määrata madalama prioriteetide hulka.

Lõputöös suudeti luua töötav raamistik eraldiseisvate moodulitena koos algoritmide ja masinõppe mudelitega, loodi mitme kategooria põhine *f-score* tulemuse hindamine ning

visuaalne tagasiside veebirakenduse kaudu. Kõige enam annab parendada mudelite tulemusi, milleks antud rakendus loodi. Kuigi mudelite tulemused ei ole perfektsed, siis selles osas oli juba varakult aimdus, et väga head tulemust ei ole võimalik saada, sest üsna varakult saadi selgeks, et sobivaid treeningandmeid leida ei ole. Samas suudeti tõestada, et luues isetreenitud mudel koos autori poolt märgendatud väikse koguse treeningandmetega, on võimalik saada arvestatav ja isegi parem tulemus mõnest eeltreenitud mudelist. See annab lootust, et lisades tulevikus juurde rohkem treeningandmeid, saab suure tõenäosusega praeguseid isetreenitud mudelite tulemusi oluliselt paremaks.

Kasutatud kirjandus

[1] T. Tammet, A. Luberg, ja P. Järv, „Sightsmap: Crowd-Sourced Popularity of the World Places“, *Inf. Commun. Technol. Tour. 2013*, jaan 2013, doi: 10.1007/978-3-642-36309-2_27.

[2] D.-D. Nguyen, M. Dao, ja T.-V. T. Nguyen, „Natural Language Processing for Social Event Classification“, *Adv. Intell. Syst. Comput.*, kd 326, lk 79–91, jaan 2015, doi: 10.1007/978-3-319-11680-8_7.

[3] E. Spyrou ja P. Mylonas, „Placing User-Generated Photo Metadata on a Map“, *2011 Sixth International Workshop on Semantic Media Adaptation and Personalization*, 2011, lk 79–84, doi: 10.1109/SMAP.2011.16.

[4] E. Spyrou ja P. Mylonas, „Analyzing Flickr metadata to extract location-based information and semantically organize its photo content“, *Neurocomputing*, kd 172, lk 114–133, jaan 2016, doi: 10.1016/j.neucom.2014.12.104.

[5] „World explorer | Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries“. https://dl.acm.org/doi/abs/10.1145/1255175.1255177.

[6] S. Qaiser ja R. Ali, „Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents“, *Int. J. Comput. Appl.*, kd 181, juuli 2018, doi: 10.5120/ijca2018917395.

[7] H. Christian, M. P. Agus, ja D. Suhartono, „Single Document Automatic Text Summarization using Term Frequency-Inverse Document Frequency (TF-IDF)“, *ComTech Comput. Math. Eng. Appl.*, kd 7, nr 4, lk 285–294, dets 2016, doi: 10.21512/comtech.v7i4.3746.

[8] J. Lilleberg, Y. Zhu, ja Y. Zhang, „Support vector machines and Word2vec for text classification with semantic features“, *2015 IEEE 14th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC)*, 2015, lk 136–140, doi: 10.1109/ICCI-CC.2015.7259377.

[9] N. Liu, C. Feng, S. Wu, A. Chan, ja J. Fulton, „Automate RFP Response Generation Process Using FastText Word Embeddings and Soft Cosine Measure“, *Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science - AICS 2019*, Wuhan, Hubei, China, 2019, lk 12–17, doi: 10.1145/3349341.3349362.

[10] E. Hindocha, V. Yazhiny, A. Arunkumar, ja P. Boobalan, „Short-text Semantic Similarity using GloVe word embedding“, kd 06, nr 04, lk 6, 2019.

[11] J. Hirschberg ja C. D. Manning, „Advances in natural language processing“, *Science*, kd 349, nr 6245, lk 261–266, juuli 2015, doi: 10.1126/science.aaa8685.

[12] A. Sun, „Short Text Classification Using Very Few Words“, lk 2.

[13] I. Pillai, G. Fumera, ja F. Roli, „Designing multi-label classifiers that maximize F measures: State of the art“, *Pattern Recognit.*, kd 61, lk 394–404, jaan 2017, doi: 10.1016/j.patcog.2016.08.008.

[14] H. van Vliet ja A. Tang, „Decision making in software architecture“, *J. Syst. Softw.*, kd 117, lk 638–644, juuli 2016, doi: 10.1016/j.jss.2016.01.017.

[15] C. Anderson, „Docker [Software engineering]“, *IEEE Softw.*, kd 32, nr 3, lk 102-c3, mai 2015, doi: 10.1109/MS.2015.62.

[16] J. Pindis ja M. Rebane, „Laserskänneriga kogutud 3D punktipilve töötlemine ja visualiseerimine“, juuni 2018. [Online]. Available at: https://digikogu.taltech.ee/et/item/09e1e129-812f-47b5-95bb-29b999015081.

[17] Y. Man, „Feature Extension for Short Text Categorization Using Frequent Term Sets“, *Procedia Comput. Sci.*, kd 31, lk 663–670, jaan 2014, doi: 10.1016/j.procs.2014.05.314.

[18] J. Chen, Y. Hu, J. Liu, Y. Xiao, ja H. Jiang, „Deep Short Text Classification with Knowledge Powered Attention“, *ArXiv190208050 Cs*, veebr 2019. [Online]. Available at: http://arxiv.org/abs/1902.08050.

[19] J. Zeng, J. Li, Y. Song, C. Gao, M. R. Lyu, ja I. King, „Topic Memory Networks for Short Text Classification“, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018, lk 3120–3131, doi: 10.18653/v1/D18-1351.

[20] D. Cer *et al.*, „Universal Sentence Encoder“, *ArXiv180311175 Cs*, apr 2018. [Online]. Available at: http://arxiv.org/abs/1803.11175.

[21] J. Pennington, R. Socher, ja C. Manning, „Glove: Global Vectors for Word Representation“, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, lk 1532–1543, doi: 10.3115/v1/D14-1162.

[22] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, ja A. Joulin, „Advances in Pre-Training Distributed Word Representations“, *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[23] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, ja T. Mikolov, „Learning Word Vectors for 157 Languages“, *ArXiv180206893 Cs*, märts 2018. [Online]. Available at: http://arxiv.org/abs/1802.06893.

[24] T. Scheepers, „Improving the Compositionality of Word Embeddings“, Master’s Thesis, Universiteit van Amsterdam, Science Park 904, Amsterdam, Netherlands, 2017.

Lisa 1 – Panoramio pildiandmete CSV lõik

Allpool on toodud lühike lõik Panoramio pildiandmete CSV failist.

```
1416486,50.918856,0.981410,36172282,4547795,noisemaker_21,2010-05-
30,"Dungeness - ""Just resting his feet"""
1416487,50.955669,0.958031,56491678,3334796,Arnand,2011-07-29,Greatstone
Sound Wall
1416488,50.943745,0.942957,56491625,3334796,Arnand,2011-07-29,Dungeness Water
Works Water Tower
1416489,50.914549,0.970241,12070095,1998366,mlriley,2008-07-13,Southern Maid
1416490,50.918822,0.981786,36147703,4495523,Caledonia,2010-05-30,3 boats
beached at Dungeness
1416491,50.942150,0.942099,56491580,3334796,Arnand,2011-07-29,Dungeness
National Nature Reserve
1416492,50.918343,0.981742,55749968,5987351,paulj8597,2011-07-15,Lou and his
Codling
1416493,50.918352,0.981627,43683857,5205319,NikkiNorman123,2010-11-
14,Skippers Boat
1416494,50.918347,0.981758,43683898,5205319,NikkiNorman123,2010-11-14,Brian
and his 8lb Bass
1416495,50.918347,0.981785,43683872,5205319,NikkiNorman123,2010-11-14,After
the fishing trip
```

Lisa 2 – Logimine

Masinõppe ja API server moodulitele on lisatud Pythoni logimine mida on võimalik lisada soovitud kohtadesse. Logimist on võimalik suunata konfiguratsioonifaili abil nii konsooli kui faili ja määrata logimise taset. Allpool toodud näide logiridadest mis sisaldavad kuupäeva ja kellaaega, Pythoni faili nime kus logirida tekitati, rea numbrit kus logirida tekitati, logimise taset ja logimise sõnumit.

```
2020-02-21 19:25:54,844 FileMapper.py 33 INFO - 0.001875162124633789 s
(convert csv to Photos)
2020-02-21 19:25:54,931 PhotosTestRepository.py 27 INFO - 0.08603143692016602
s (insert to db)
2020-02-21 19:25:54,950 PhotosTestRepository.py 33 INFO -
0.019138574600219727 s (select from db)
2020-02-21 19:25:54,961 PhotoClusterer.py 37 INFO - Photos left to cluster:
58
2020-02-21 19:25:54,961 PhotoClusterer.py 37 INFO - Photos left to cluster:
57
2020-02-21 19:25:54,963 PhotoClusterer.py 37 INFO - Photos left to cluster:
33
2020-02-21 19:25:54,963 PhotoClusterer.py 37 INFO - Photos left to cluster:
26
2020-02-21 19:25:54,963 PhotoClusterer.py 37 INFO - Photos left to cluster:
25
2020-02-21 19:25:54,963 PhotoClusterer.py 37 INFO - Photos left to cluster:
23
```

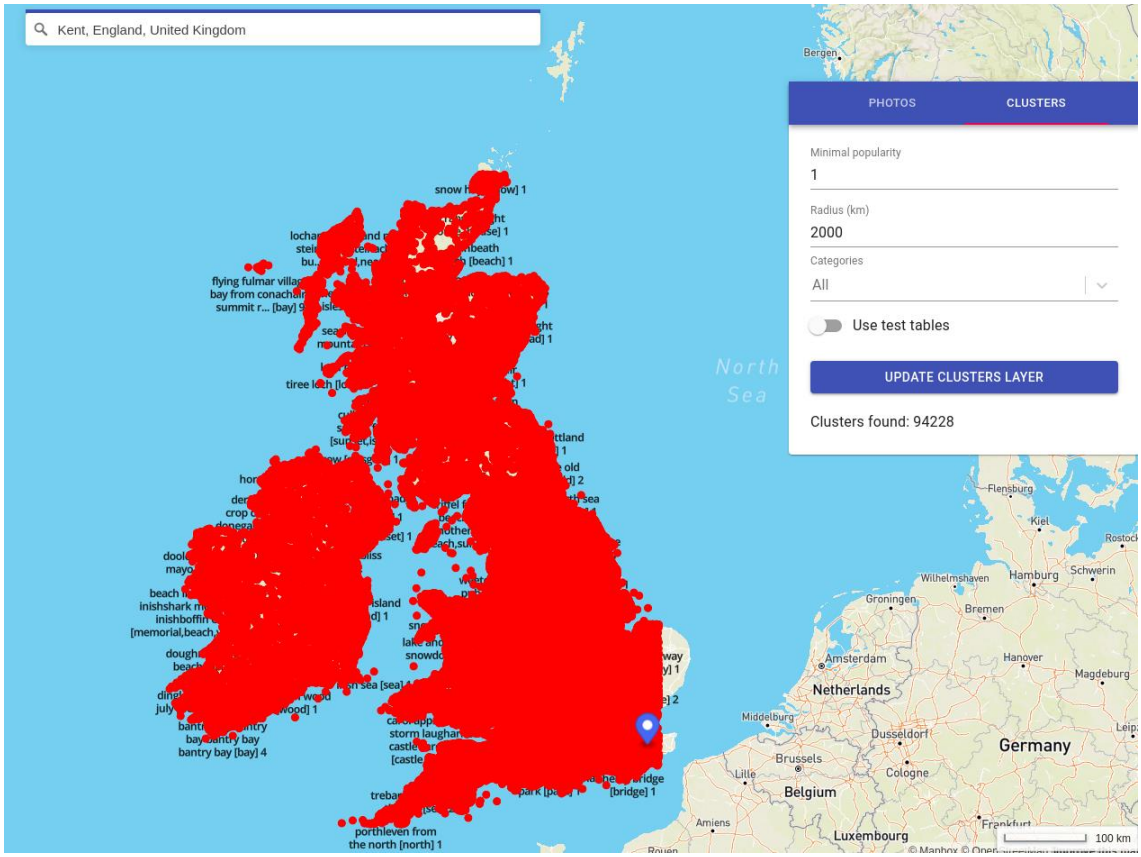
Lisa 3 – Pildiandmete koondamise kiiruse optimeerimine

Töö lõppfaasis prooviti kiirendada pildiandmete kogumiteks koondamise etappi. Suudeti luua lahendus, mis viis koondamise etapi 53 tunnilt 30 minuti peale. Selleks lisati juurde pildiandmete sorteerimine pikkuskraadide alusel, mis võimaldas vähendada pildiandmete kandidaatide hulka, mida prooviti loodavasse kogumisse lisada.

Kiiremat lahendust antud lõputöös siiski reaalselt enam ei rakendatud, kuna uue algoritmi loodud kogumid erinesid varem loodud kogumitest, mille seast olid valitud käsitsi test -ja treeningandmete kogumid. Erinevused tulenesid peamiselt sellest, et ühel juhul olid enne koondamist pildid sorteeritud koordinaatide alusel ja teisel juhul mitte. See, et uue kiirema algoritmi tulemused olid erinevad, ei tähendanud, et need olid valed, vaid need olid loodud teistsuguse loogika alusel. Kiirem lahendus võib tulla kasuks tulevikus, kui lisatakse juurde uusi test -ja treeningandmeid või näiteks täiesti uute pildiandmete kategoriseerimisel.

Lisa 4 – Kogumite geograafiline kaetus

Alloleval joonisel on näidatud mis geograafilises piirkonnas on kogumid loodud. Panoramio algandmetest olid Inglismaa idaranniku äärsed punktid puudu. Seda on näha samuti allolevalt kogumite jooniselt.



Vaade kõikide kogumite geograafilisest kaetusest.