Uniwersytet Przyrodniczy we Wrocławiu
Wydział Inżynierii Kształtowania Środowiska i Geodezji
Kierunek: Geodezja i kartografia
Przedmiot: Metody Eksploracji Danych

Wrocław, 02.02.2015

SPRAWOZDANIE Z WYKONANYCH PRAC

WYKRYJ CENTRUM MIASTA WEDŁUG ZAGĘSZCZENIA PRZYSTANKÓW KOMUNIKACJI MIEJSKIEJ (OPENSTREETMAP)

SEKCJA:
TOMASZ KRAWCZYK
ELŻBIETA LASOTA
KRZYSZTOF SOCHIERA
GiK I MSU, gr. 4
Rok akademicki 2014/2015

SPIS TREŚCI

Wstęp

Wynikiem ćwiczenia jest wtyczka do programu QGIS. Proces stworzenia wtyczki był złożony. Na jego działanie składa się pobranie danych z Open Street Map. Zaimportowanie ich do bazy MongoDB, a następnie import do skryptu pythonowego zapisanego w formie wtyczki do programu QGIS.

1. Openstreetmap.org

   Jest to portal gromadzący geodane. Działa w przeglądarce internetowej. Jest całkowicie darmowy. W przeciwieństwa do najpopularniejszych serwisów mapowych, Openstreetmap (OSM) jest tworzony przez zarejestrowanych użytkowników-wolontariuszy. Na podstawie dostępnych starych map, zdjęć satelitarnych i własnej wiedzy, użytkownicy edytują mapę i tworzą wielki, ciągle aktualizowany projekt. Użytkownicy tworzą społeczność, która corocznie spotyka się na konferencji.

   Pomysłodawcą tego serwisu jest Steve Coast, który wzorował się na Wikipedii. Podobnie jak ta największa encyklopedia internetowa, dane są dostępne dla wszytkich za darmo.

   Dane dostępne w OSM można również pobierać na dysk twardy. W samym serwisie istnieje opcja Export. Gdy potrzeba wszystkich danych OSM, można pobrać uajtualnia raz w tygodniu plik, którego rozmiar wynosi ok. 498 GB. Do tego służy narzędzie Planet.osm. Istnieje również mniejsza wersja zawierająca dane dla poszczególnych kontynentów, państw i regionów nazwana Extracts.

   Oprócz Plantet.osm istnieje wiele serwisów zewnętrznych oferujących dane przez protokół http lub przez protokół FTP. Bardzo znanym przykładem takiej witryny jest geofabrik.de obsługująca dane dla całej Europy oraz wielu innych państw świata.

   Nasze dane zostały pobrane przez witrynę Overpass. Jest to witryna internetowa odwiedzana codzienne przez ponad milion osób. By skorzystać z danych należy ułożyć zapytanie zgodnie z napisanym wcześniej API. Jest to szybkie narzędzie pozwalające pobrać nie tylko surowe dane z OSM, ale także je odpowiednio odfiltrować, np. wybrać tylko interesującą nas warstwę. W naszym przypadku była to warstwa public_transport dla miasta Wrocław (a później także dla innych miast). Otrzymaliśmy dane w formacie JSON.

2. MongoDB

   Jest to otwarty system zarządzania bazą danych(SZBD). Jego pierwsza stabilna wersja ukazała się roku 2009. Od tego czasu jest rozwijana przez firmę MongoDB. Istnieje duża społeczność pomagająca w pracach nad tym projektem.

   MongoDB działa nie w oparciu o język zapytań SQL, jak najpopularniejsze w tej chwili SZBD. Zamiast tego promuje rozwiązanie ad-hoc napisane w języku javaScript. Dużą zaletą tego programu jest elastyczna budowa relacji. Poszczególne krotki nie muszą mieć jednolitej budowy tak jak np. w MS SQL Server czy postgreSQL. Każdy dokument przypomina plik JSON. Wymiana danych z innymi aplikacjami jest bardzo łatwa. Wystarczy kilka kliknięć by importować bądź eksportować dane. SZBD wspiera także funkcje agregujące oraz indeksowanie. Wśród wad wymienia się brak kodowania UTF-8, co w znacznie utrudnia działanie aplikacji, gdy dane są w języku innym niż angielski. Problemem są także transakcje. Są one zawężone do jednego dokumentu. Dużą zaletą jest natomiast kooperacja z system Linux. Wszystkie komendy, takie jak import danych czy przeglądanie danych można wpisać z terminala komend pod każdą dystrybucją Linuxa.

   Dane pobrane z OSM były zapisane w formacie JSON, który jest wspierany przez

MongoDB. Jedynym dobrym problemem był nagłówek w pliku z danymi, który początkowo utrudniał import danych. Po wyeliminowaniu tego problemu, import przebiegł poprawnie.

3. Scikit learn

To otwarta biblioteka dla Pythona. Zawiera wiele zaimplementowanych metod klasyfikacji, regresji i klasteryzacji. Wspiera wektory regresji, algorytm naiwny Bayesa, funckję k-means rad dbscan.. Jest także zaprojektowany do współpracy z innymi modułami naukowymi w pythonie, takimi jak NumPy oraz SciPy. Początkowo był on tylko rozszerzeniem do SciPy, ale z czasem wyodrębnił się w osobny moduł.

Dwie najczęściej spotykane metody klasteryzacji to k-means oraz dbscan. K-mean, po polsku algorytm centroidów, jest wykorzystywany również w kwantyzacji wektorowej. Natomiast dbscan jest algorytmem opartym na gęstości punktów. Jest jednym z najpowszechniej używanych algorytmów w nauce, a artykuł, w którym został opisany jest bardzo często cytowany. Ma szeroki wachlarz zastosowań, również w geodezji oraz metodach eksploatacji danych. Ciekawostką jest fakt, że został nagrodzony w czasie konferencji KDD, największej poświęconej eksploracji danych.

Instalacja tego modułu może być trochę kłopotliwa. Program wymaga być wcześniej zainstalowano dwa inne dodatki: NumPy oraz SciPy.

4. QGIS

Jest programem gisowym. Rozwijany jest przez społeczność zgromadzoną wokół projektu Open Source Geospatial. Jest poważną alternatywą dla komercyjnych programów typu ArcGIS i Geomedia. Wśród jego możliwości jest także obsługa plików z głównych programów gisowych na rynku. Przy instalacji tego programu instalowane są jeszcze inne programy gisowe, takie jak SagaGIS oraz GrassGIS. Dla tego drugiego pełni nawet rolę interfejsu graficznego. Pozwala na współpracę z danych z baz danych, np. postgreSQL. Import i pracę na warstwach wektorowych oraz rastrowych umożliwiają biblioteki GDAL i OGR. Wśród innych możliwości tego programu są analizy przestrzenne oraz analizy rastrowe.

Bardzo dużą zaletą tego systemu jest możliwość rozwijania jego funkcjonalności przez tworzenie i instalację wtyczek. QGIS jest projektem otwartych, dlatego społeczność jest mocno zaangażowana w jego współtworzenie, dlatego w internecie znajdziemy pluginów gotowych do instalacji.

Znajomość działania moduły Pythona o nazwie PyQGIS pozwala na samodzielne tworzenie skryptów i automatyzację pracy w tym programie. Można wykorzystać to na kilka sposobów. Najbardziej elementarnych sposobem jest wpisywanie poleceń do konsoli. Do niej jest dołączony prosty edytor tekstu. Można wpisać do niego serię instrukcji, które zostaną wykonane po wczytaniu i uruchomieniu skryptu. Trzecią, najbardziej zaawansowaną możliwością jest tworzenie własnych wtyczek.

Implementację własnych wtyczek ułatwiają… inne wtyczki. Przykładowo wtyczka Plugin Builder pozwala na rozpoczęcie pracy z programowaniem tworząc podstawowe i wymagane do działania pliki, zawierające miedzy innymi metadane oraz główny skrypt. Zaletą plugina nad innymi zastosowaniami programowania w tym programie jest możliwość dodania interfejsu do skryptu, a co za tym idzie optymalizacja i większy wpływ użytkownika w jego działanie.

Opis prac

Jako temat projektowy wybraliśmy dostępny na liście temat nr 2 pt: „Wykryj centrum miasta według zagęszczenia przystanków komunikacji miejskiej (OpenStreetMap lub jakdojade.pl)”.

Pierwszy etap prac rozpoczęliśmy od przygotowania danych do analizy. Jako dane wyjściowe postanowiliśmy wykorzystać dane z OpenStreetMap. Dane można pobrać bezpośrednio

api OSM bądź z xapi, który jest rozbudowaną wersją klasycznego api i możliwa rozszerzone szukanie i stosowanie zapytań.

Docelowo skorzystaliśmy z Overpass turbo, które udostępnia wybrane części mapy OSM. Jest to idealne narzędzie dla eksploracji danych. Dostępne jest pod adresem: http://overpass-turbo.eu/ .

Na stronie istnieje osobne okno do tworzenia zapytań i kreator ułatwiający ich tworzenie. Wyszukane dane można pobrać, a samo zapytanie zapis w bazie w razie potrzeby, gdy będzie nam potrzebne później. Dla naszego tematu postanowiliśmy wyszukać wszystkie przystanki publicznego transportu na terenie Wrocławia. W tym celu na początku skorzystaliśmy ze strony http://wiki.openstreetmap.org/wiki/Map_Features , aby zapoznać się z typami obiektów OSM i znaleźć oznaczenie przystanków. Oznaczenie to znaleźliśmy w sekcji Public Transport, gdzie klucz "public_transport" przyjmuje wartość "stop_position". Na podstawie tych danych stworzyliśmy zapytanie w Overpass-turbo, dodając iż obszarem zapytania jest Wrocław:

```
[out:json][timeout:25];
{{geocodeArea:Wroclaw}}->.searchArea;
(
  node["public_transport"="stop_position"](area.searchArea);
  way["public_transport"="stop_position"](area.searchArea);
  relation["public_transport"="stop_position"](area.searchArea);
);
out body;
>;
out skel qt;
```

Wyszukane dane można ściągnąć na dysk w kilku dostępnych formatach. Wśród nich znajduje się również geojson, który idealnie odpowiada naszym wymaganiom.

Przystanki w formie geojson następnie zaimportowaliśmy do bazy danych. Najbardziej odpowiednią i elastyczną bazą danych dla tego typu prac, które prowadzimy jest baza danych MongoDB. Jednak nasz kontakt z nią dotychczas był bardzo ograniczony i pojawił się problem z importem danych do niej. Jednak dzięki pomocy na zajęciach udało się problem rozwiązać. W ten sposób mamy bazę danych z zaimportowanymi przystankami. Następnie utworzyliśmy indeks przestrzenny na geometrii poleceniem:
db.collection.ensureIndex({ <location field> : "2dsphere" }”),
dzięki któremu możemy operować na operacjach przestrzennych.

Jako iż w dalszej pracy będziemy korzystać z języka skryptowego Python i dostępnych tam bibliotek do klasteryzacji. Wyszukaliśmy bibliotekę Pythona do obsługi bazy MongoDB. W tym celu ściągnęliśmy i zainstalowaliśmy pakiet PyMongo a następnie utworzyliśmy połączenie z bazą danych:

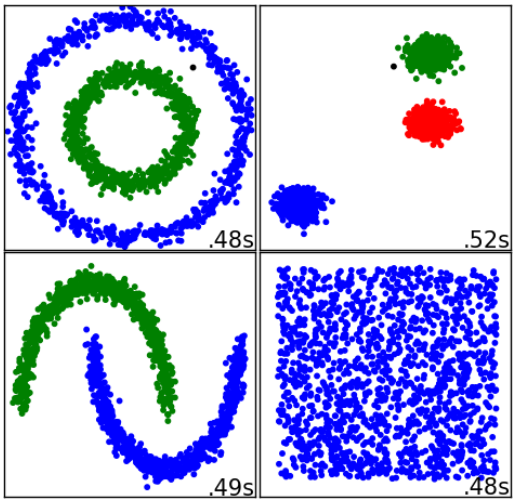
```
client=MongoClient()          #połączenie z klientem
db=client.MED                 #połączenie z bazą danych
collection=db.przystanki      #połączenie z kolekcją zawierającą przystanki
```

W następnym kroku zaczęliśmy poszukiwać bibliotek i algorytmów klasteryzacji, które podzielą nam obszar miasta na skupiska przystanków. Wykorzystaną w tym celu biblioteką jest scikit-learn. przypadku scikit-learn znaleźliśmy ciekawą stronę dotyczącą klasteryzacji: http://scikit-learn.org/stable/modules/clustering.html.

Algorytm zaimplementujemy został jako wtyczka do QGISa z, co pozwoliło dalej na bardzo dobrą wizualizację wyników.

DBScan

Biblioteka scikit-learn udostępnia szereg algorytmów klasyfikacji. Jedną z nich jest DBScan. Może on być wykorzystany dla geometrii płaskich. Jeśli chodzi o wymiary klastra to mogą być one różnej wielkości. Natomiast wykorzystywaną w tym algorytmie miarą jest odległość punktów najbliższych.



Rys. 1. Przykładowe wyniki działania algorytmu DBScan

Algorytm DBScan możemy przedstawić następująco [1]:

- Przez sąsiedztwo wektora danych x rozumiemy zbiór
  $\{y \in D : d(x, y) < \varepsilon\}$,
  gdzie wartość ε jest parametrem algorytmu DBScan.
- Sąsiedztwo wektora danych x jest gęste, jeśli zawiera co najmniej m wektorów danych, gdzie wartość m jest parametrem algorytmu DBScan.
- Rdzeń to wektor danych, którego sąsiedztwo jest gęste.
- Punkt brzegowy to wektor danych, którego sąsiedztwo nie jest gęste.
- Wektor danych y jest bezpośrednio osiągalny z wektora danych x, jeżeli y należy do sąsiedztwa x oraz sąsiedztwo x jest gęste.
- Wektor danych y jest osiągalny z wektora danych x, jeśli istnieje ciąg wektorów danych x1, x2, …, xn, taki że x1 = x, xn = y oraz xi jest bezpośrednio osiągalny z xi-1, dla każdego i = 2, 3, …, n.
- Wektory danych x i y są połączone, jeśli istnieje wektor danych z, taki że x i y są osiągalne z z.
- Grupa to maksymalny zbiór punktów połączonych.
- DBScan:
  - wybierz dowolny nierozpatrzony jeszcze wektor danych x
  - oznacz x jako już rozpatrzony
  - C := zbiór wszystkich wektorów danych osiągalnych z x
  - jeśli x jest rdzeniem, to uznaj C za grupę i oznacz wszystkie elementy C jako już rozpatrzone
  - powtarzaj powyższe kroki aż wszystkie wektory danych zostaną rozpatrzone.

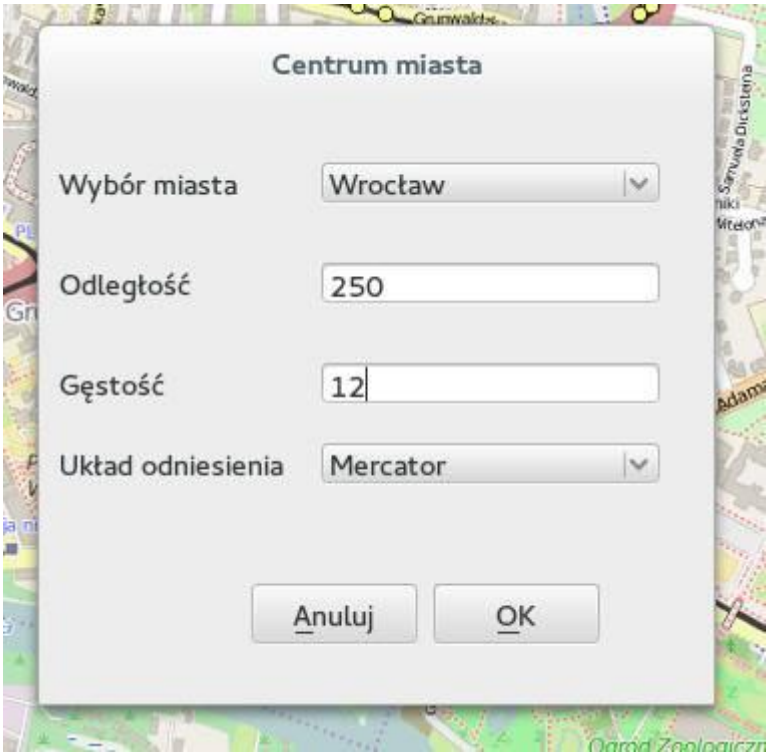
Przykładowe wyniki działania dla różnych zbiorów danych przedstawia Rys. 1.

Testy

Dostępne metody klas dostępnych w bibliotece scikit-learn pozwoliły na zaimplementowanie w kodzie programu języka Python funkcji, które pozwalają na podstawie podanych przez użytkownika, w graficznym interfejsem użytkownika wtyczki QGIS-a, parametrów (Rys. 2). Pierwszy parametr pozwala określić dla jakiego miasta z listy wykonywane są analizy. Dodatkowo wśród należy wprowadzić te związane z wykorzystanym algorytmem klasteryzacji – DBSCAN, który wymaga podania parametrów takich jak odległość odpowiadającej długości promienia określającego sąsiedztwo i gęstość mówiący o liczbie obiektów danych stanowiąca o gęstości sąsiedztwa. Istnieje także możliwość wyboru układu odniesienia, który domyślnie ustawiony jest na odwzorowanie Mercatora.

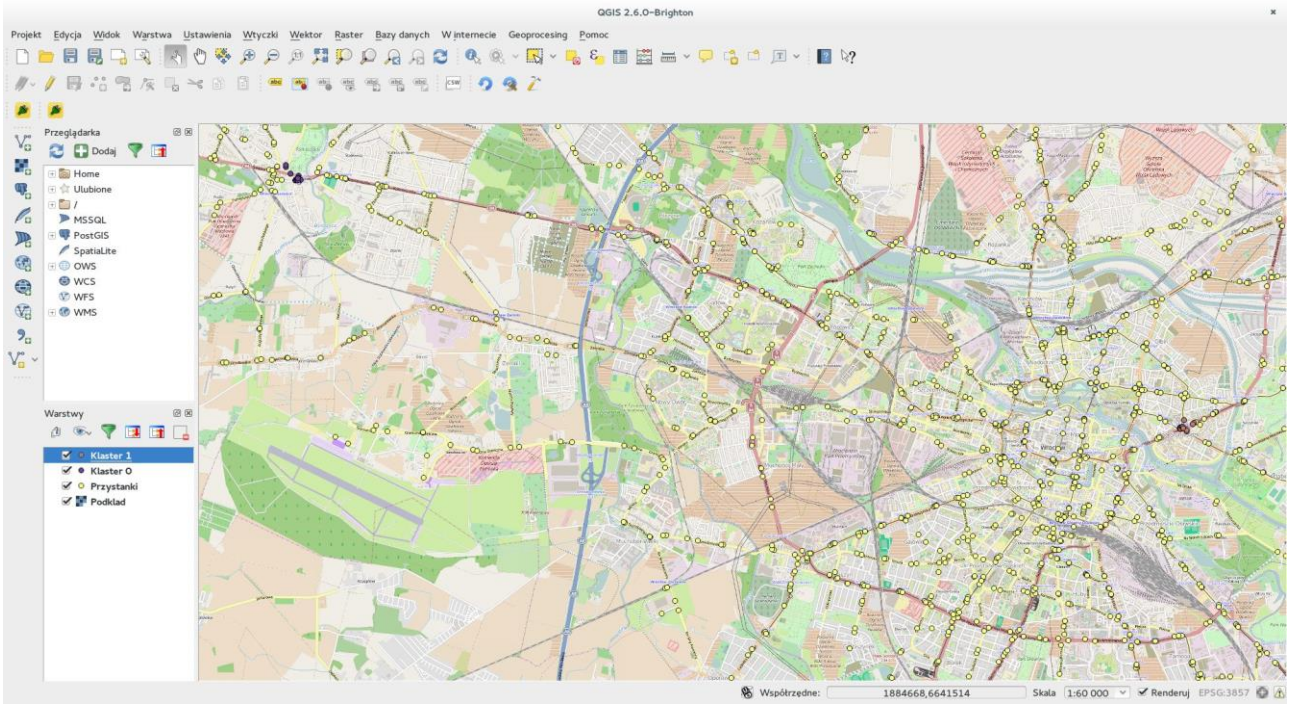
W wyniku przeprowadzonych testów, metodą prób i błędów określono, że najoptymalniejszymi wartościami parametrów dla algorytmu DBScan dla obszaru Wrocławia są:

- Odległość – 250 m,
- Gęstość – 12 punktów.

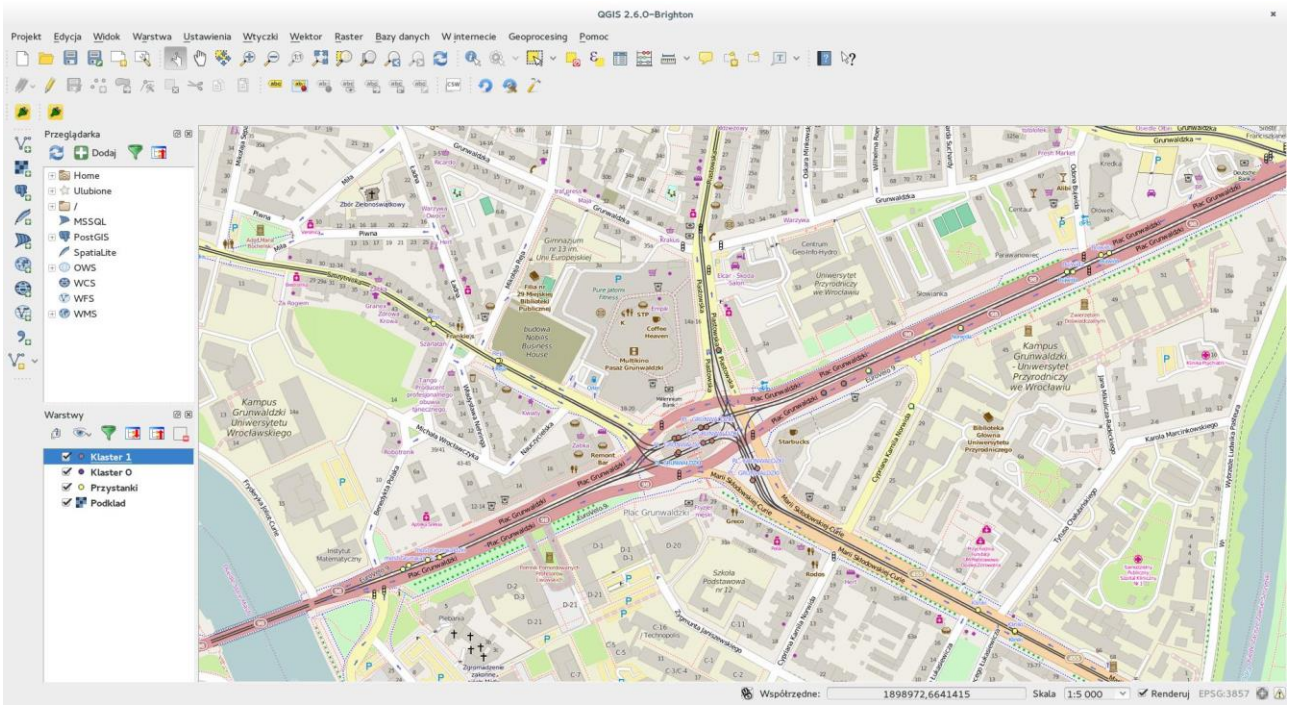


Rys. 2. Graficzny interfejs użytkownika zaimplementowanej wtyczki

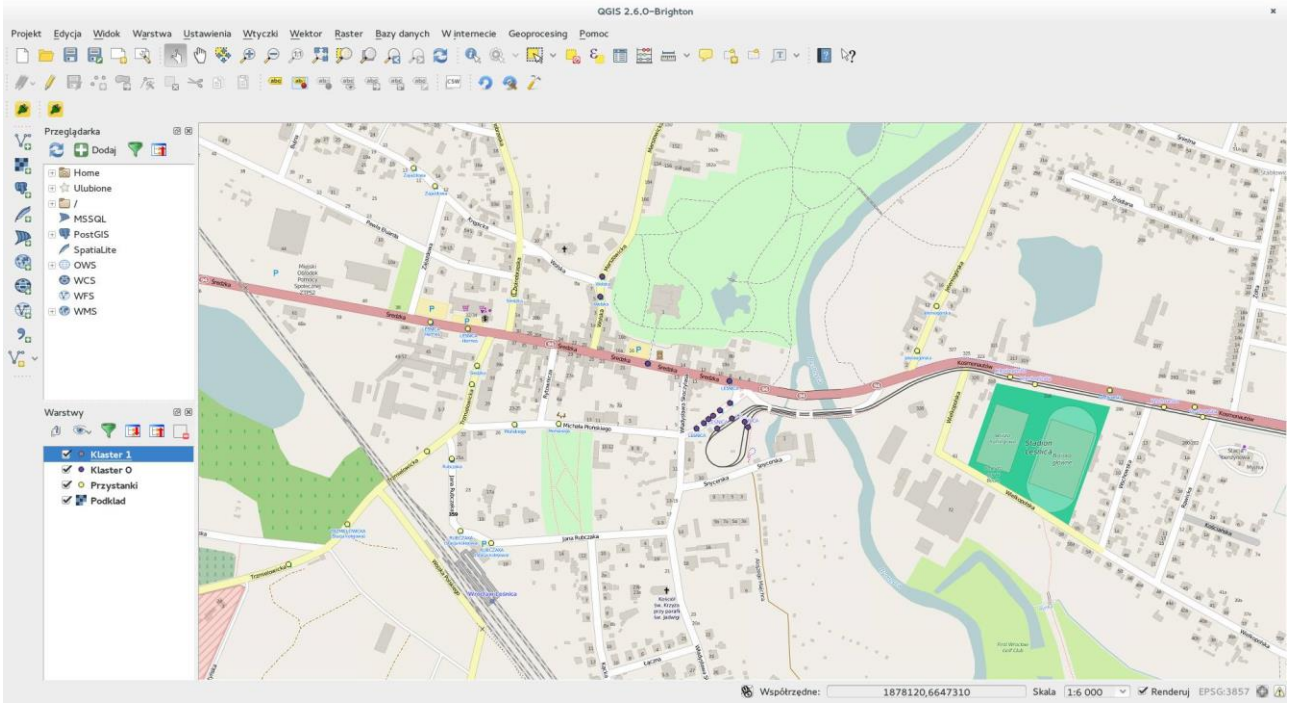
Pierwszym obszarem, dla którego przeprowadzone zostały testy był teren Wrocławia. W wyniku działania programu zostały wydzielone dwa klastry reprezentujące centrum miasta (Rys. 3). O ile jeden z nich jest położony w niedalekiej odległości od rynku, a dokładniej na rondzie Reagana (Rys. 4), to drugi jest położony na Leśnicy, która znajduje się w znacznej odległości od rynku. Może być to spowodowane rozmieszczeniem punktów na terenie Leśnicy, gdzie na niewielkim obszarze znajduje się ich 10 (Rys. 5). Jest to jednak niezgodne ze stanem faktycznym, gdyż znajduje się ich tam zdecydowanie mniej. Problem ten pozwoliło by pobranie danych z innego źródła. Jednak dotarcie do danych takich jak dane portalu jakdojade.pl albo MPK we Wrocławiu jest problematyczne lub wręcz niemożliwe.



Rys. 3. Wyniki działania dla obszaru Wrocławia (odległość – 250 m, gęstość – 12 punktów)

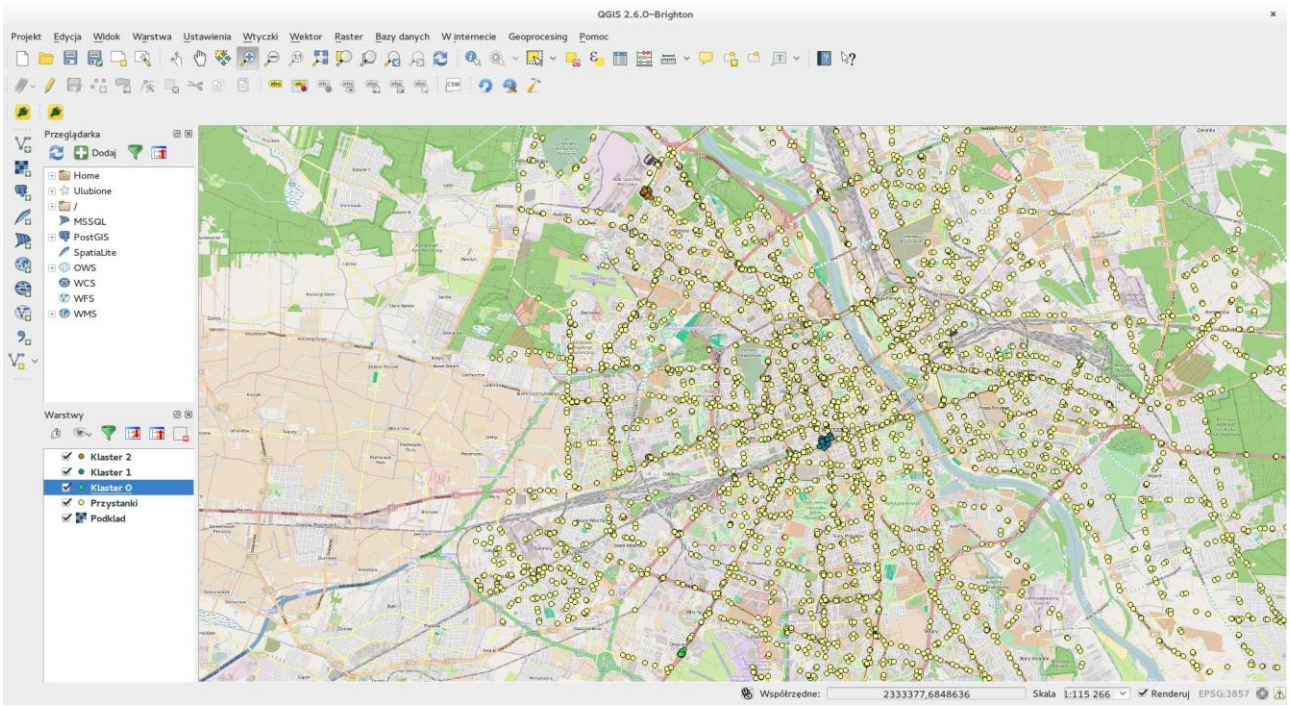


Rys. 4. Klaster w okolicy ronda Reagana

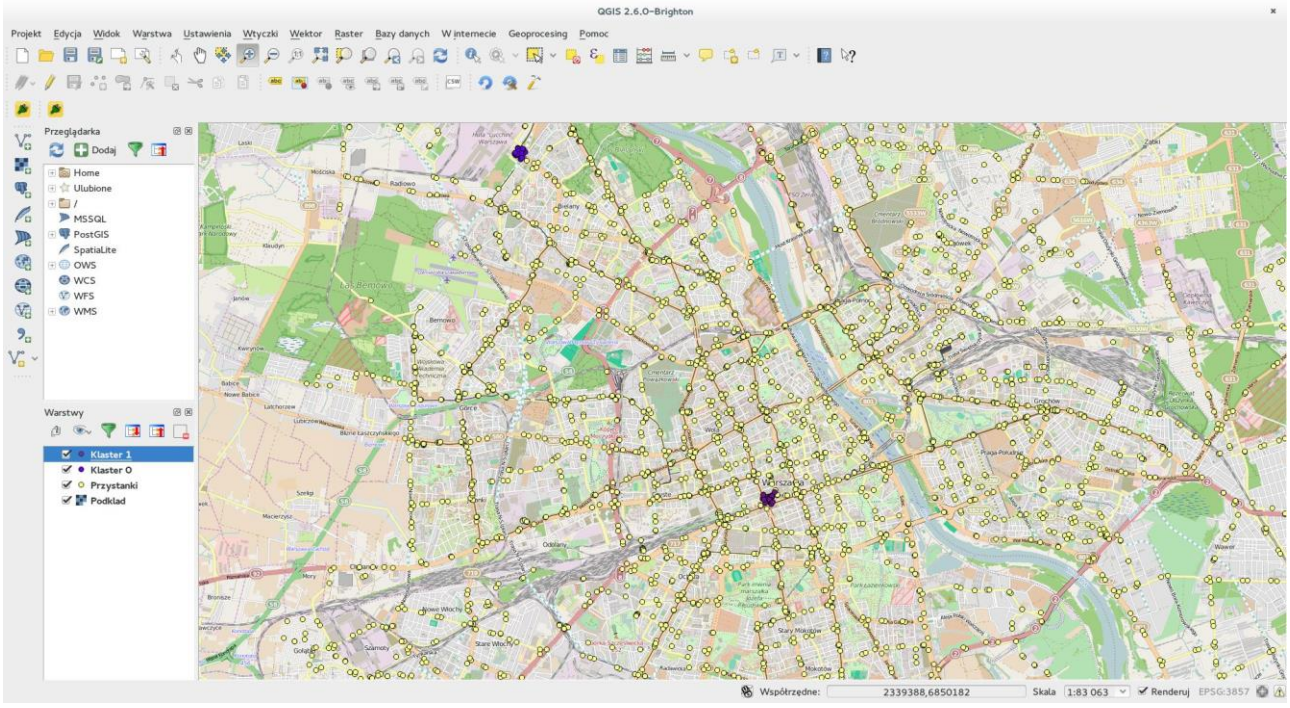


Rys. 5. Klaster w okolicy Leśnicy

Kolejnym miastem, które podlegało analizie jest Warszawa, dla której zastosowano dwa różne zestawy parametrów. Dla pierwszego z nich liczba klastrów wyniosła również 2. Jeden z nich zlokalizowany został w okolicach zajezdni Żoliborz, która także jak w przypadku Leśnicy jest znacznie oddalona od faktycznego centrum miasta (Rys. 6). Znacznie lepiej została wyznaczona druga klasa, która została określona praktycznie w samym centrum miasta. Drugi zbiór parametrów dał takie same wyniki uzupełnione o jeszcze jedną klasę położona w okolicach Okęcia (Rys. 7).

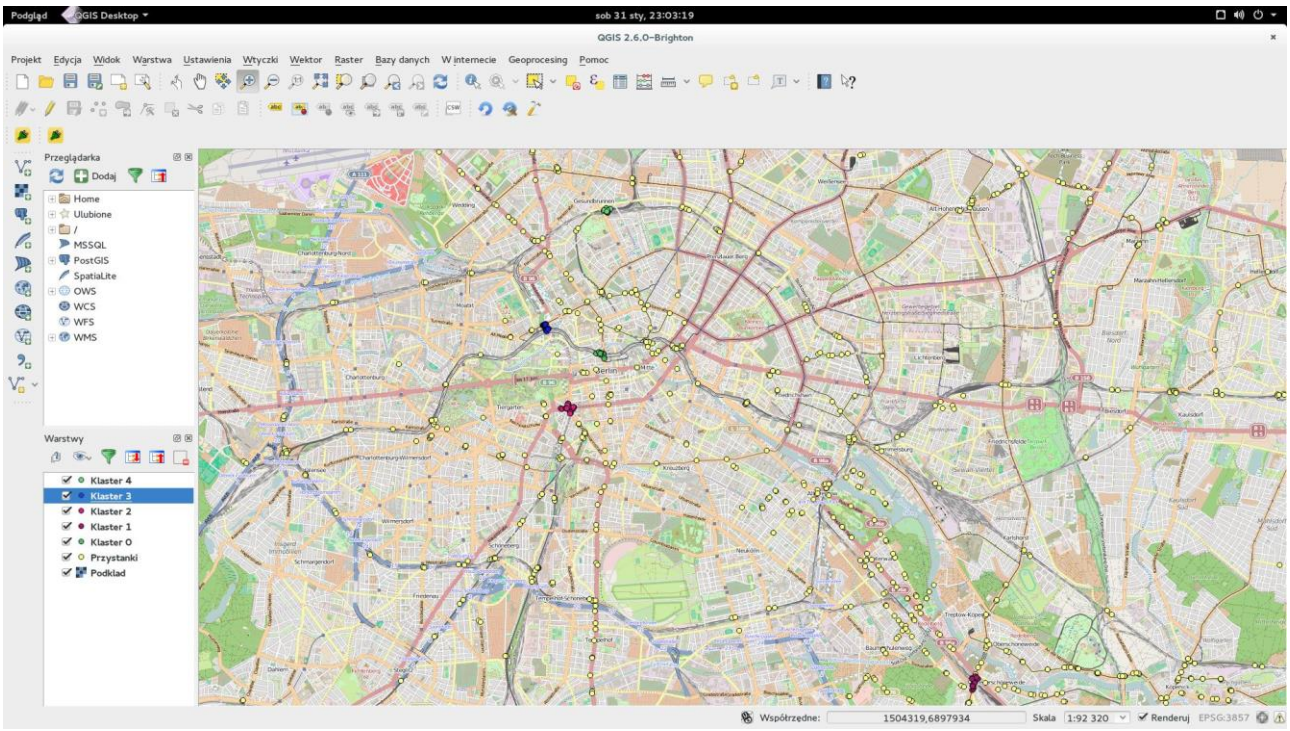


Rys. 6. Wyniki działania dla obszaru Warszawy (odległość – 200 m, gęstość – 22 punktów)

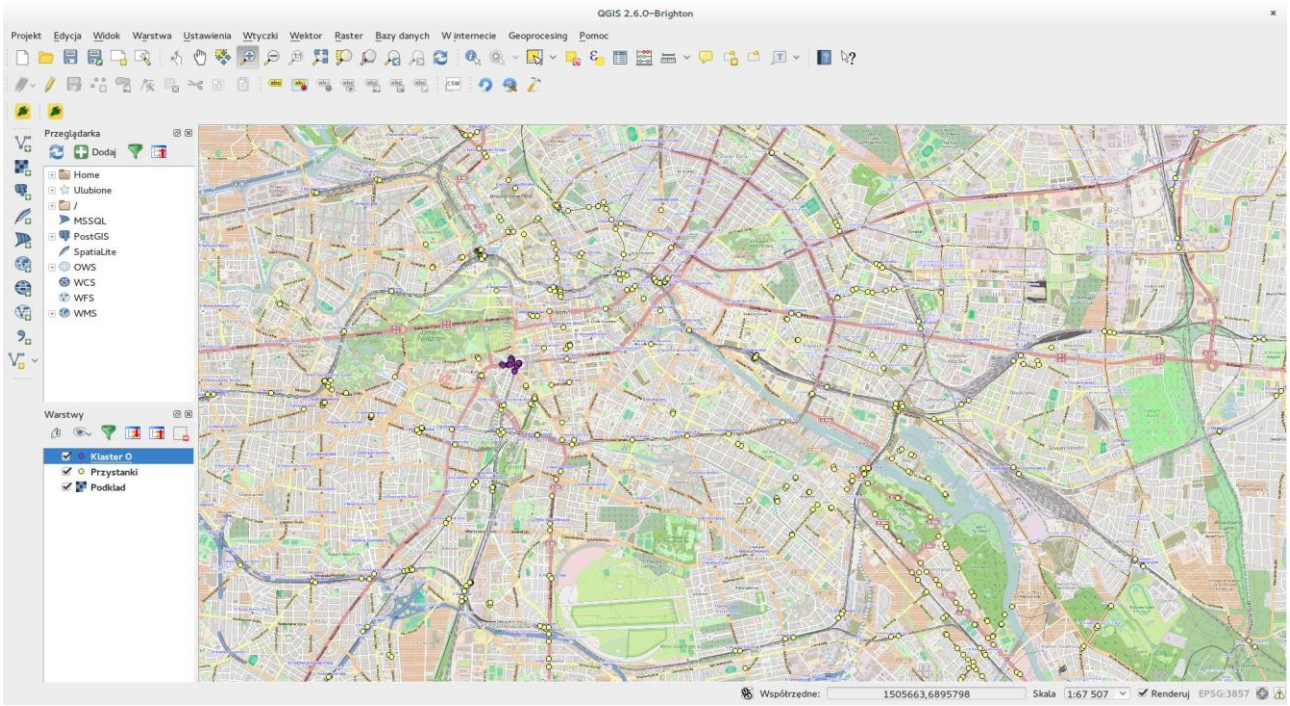


Rys. 7. Wyniki działania dla obszaru Warszawy (odległość – 250 m, gęstość – 22 punktów)

Ostatnim miastem, dla którego wyznaczono centra na podstawie przystanków jest Berlin. Dla którego niezależnie od użytych parametrów kolejne klasy znajdują się w okolicach centrum (Rys. 8, Rys. 9).



Rys. 8. Wyniki działania dla obszaru Berlina (odległość – 250 m, gęstość – 12 punktów)



Rys. 7. Wyniki działania dla obszaru Berlina (odległość – 250 m, gęstość – 18 punktów)

Bibliografia

[1] Piotr Lipiński, Wykład z eksploracji danych
[2] http://wiki.openstreetmap.org/wiki/Map_Features
[3] http://scikit-learn.org/stable/modules/clustering.html
[4] http://overpass.osm.rambler.ru/cgi/interpreter