Which Bus?
OPEN DATA CHALLENGE 2014

Άρης Φεργάδης (fergadis.aris@gmail.com)

26 Σεπτεμβρίου 2014

Περίληψη

Η εφαρμογή που υλοποιήθηκε για τον διαγωνισμό OPEN DATA CHALLENGE 2014 κάνει χρήση των ανοιχτών δεδομένων και έχει ως σκοπό την εξυπηρέτηση των πολιτών προτείνοντάς τους τις πιο κοντινές στάσεις του ΟΑΣΑ προς τη δημόσια υπηρεσία που θέλουν να επισκεφτούν. Η εφαρμογή μπορεί να χρησιμοποιηθεί είτε ως αυτόνομη, είτε ως Web Service.

Εισαγωγή

Στο πλαίσιο της προτροπής των πολιτών για χρήση των Μέσων Μαζικής Μεταφοράς θα πρέπει να υπάρχει ένας εύκολος τρόπος να βρουν ποιες λεωφορειακές γραμμές και ποιες στάσεις εξυπηρετούν τις δημόσιες υπηρεσίες. Με αυτό το σκεπτικό και στο πλαίσιο του διαγωνισμού OPEN DATA CHALLENGE 2014, υλοποιήθηκε η εφαρμογή "Which Bus?" στην οποία ο χρήστης εισάγει το όνομα της υπηρεσίας και τον αριθμό των κοντινών στάσεων. Το σύστημα εμφανίζει τα ονόματα των στάσεων και τις λεωφορειακές γραμμές που περνούν από εκεί.

Τα Δεδομένα

Η εφαρμογή χρησιμοποιεί ανοιχτά δεδομένα τα οποία προέρχονται από το geodata.gov.gr. Συγκεκριμένα, έγινε χρήση δύο σετ δεδομένων:

1. Δημόσια Κτίρια (`http://goo.gl/jNKf6l`)
    - Τελευταία Ενημέρωση: 2011-02-04
    - Άδεια: Creative Commons Αναφορά Προέλευσης (CC BY v.3.0)

    Τα δεδομένα αποτελούνται από ένα αρχείο μόνο μορφής csv.

2. Δρομολόγια των Αστικών Συγκοινωνιών της Αθήνας (`http://goo.gl/MBTSTH`)
    - Τελευταία Ενημέρωση: 2013-12-18
    - Άδεια: Creative Commons Αναφορά Προέλευσης (CC BY v.3.0)

    Τα δεδομένα αποτελούνται από τα αρχεία: `agency.txt`, `calendar.txt`, `calendar_dates.txt`, `feed_info.txt`, `routes.txt`, `shapes.txt`, `stops.txt`, `stop_times.txt`, `trips.txt`. Για την εφαρμογή χρησιμοποιήθηκαν τα: `routes.txt`, `stops.txt`, `stop_times.txt`.

Λόγω του γεγονότος ότι τα δεδομένα για τα Δημόσια Κτίρια αναφέρονται σε όλη την Ελλάδα αλλά τα δρομολόγια μόνο στις περιοχές τις Αττικής που καλύπτει ο ΟΑΣΑ, για την εφαρμογή χρησιμοποιήθηκε ένα υποσύνολο των κτιρίων και συγκεκριμένα όσα ανήκουν στην περιφέρεια Αττικής.

Η Εφαρμογή

Η εφαρμογή μπορεί να εκτελεστεί είτε ως αυτόνομη (stand alone) είτε ως Web Service δίνοντας έτσι τη δυνατότητα σε άλλες εφαρμογές να επωφεληθούν από τα αποτελέσματά της.

Stand Alone

Για χρήση ως αυτόνομη δίνουμε την εντολή `python3 which_bus.py` σε ένα τερματικό. Κατά την εκτέλεση της εφαρμογής ζητούνται δύο δεδομένα από το χρήστη: 1) το όνομα της υπηρεσίας (κτιρίου) και 2) το πλήθος των κοντινών στάσεων που θα εμφανιστούν.

Ακολουθεί ένα παράδειγμα εκτέλεσης.

```
Δώστε την ονομασία του κτιρίου (π.χ. 5ο ΓΥΜΝΑΣΙΟ ΑΘΗΝΑΣ): 5ο ΓΥΜΝΑΣΙΟ ΑΘΗΝΑΣ
Πλήθος στάσεων προς αναζήτηση (π.χ. 3): 4
Αναζήτηση Στάσης για: 5ο ΓΥΜΝΑΣΙΟ ΑΘΗΝΑΣ

Αποτελέσματα
Ονομασία Στάσης: ΜΠΟΤΑΣΗ, Επί της ΣΟΛΩΝΟΣ
    Γραμμή: 021, ΚΑΝΙΓΓΟΣ - ΓΚΥΖΗ
    Γραμμή: 060, ΜΟΥΣΕΙΟ - ΑΚΑΔΗΜΙΑ - ΛΥΚΑΒΗΤΤΟΣ (ΚΥΚΛΙΚΗ)
    Γραμμή: 224, ΚΑΙΣΑΡΙΑΝΗ - ΕΛ. ΒΕΝΙΖΕΛΟΥ

Ονομασία Στάσης: ΑΦΕΤΗΡΙΑ, Επί της ΤΖΩΡΤΖ
    Γραμμή: 021, ΚΑΝΙΓΓΟΣ - ΓΚΥΖΗ

Ονομασία Στάσης: ΖΩΟΔ.ΠΗΓΗΣ, Επί της ΑΚΑΔΗΜΙΑΣ
    Γραμμή: 022, Ν. ΚΥΨΕΛΗ - ΜΑΡΑΣΛΕΙΟΣ
    Γραμμή: 054, ΠΕΡΙΣΣΟΣ - ΑΚΑΔΗΜΙΑ - ΜΕΤΑΜΟΡΦΩΣΗ
    Γραμμή: 060, ΜΟΥΣΕΙΟ - ΑΚΑΔΗΜΙΑ - ΛΥΚΑΒΗΤΤΟΣ (ΚΥΚΛΙΚΗ)
    Γραμμή: 100, ΠΛ. ΚΟΥΜΟΥΝΔΟΥΡΟΥ - ΚΟΛΩΝΑΚΙ - ΑΓΟΡΑ (ΚΥΚΛΙΚΗ)
    Γραμμή: 224, ΚΑΙΣΑΡΙΑΝΗ - ΕΛ. ΒΕΝΙΖΕΛΟΥ
    Γραμμή: 608, ΓΑΛΑΤΣΙ - ΑΚΑΔΗΜΙΑ - ΝΕΚΡ. ΖΩΓΡΑΦΟΥ
    Γραμμή: 622, ΓΟΥΔΗ - ΑΝΩ ΓΑΛΑΤΣΙ
    Γραμμή: 732, ΑΓ. ΦΑΝΟΥΡΙΟΣ - ΑΚΑΔΗΜΙΑ - ΖΩΟΔ. ΠΗΓΗ

Ονομασία Στάσης: ΠΛ.ΚΑΝΙΓΓΟΣ, Επί της ΠΛ.ΚΑΝΙΓΓΟΣ
    Γραμμή:   6, ΙΠΠΟΚΡΑΤΟΥΣ - Ν. ΦΙΛΑΔΕΛΦΕΙΑ - ΚΟΚ. ΜΥΛΟΣ
    Γραμμή:  13, ΛΑΜΠΡΙΝΗ - ΠΛ. ΚΑΝΙΓΓΟΣ - Ν. ΨΥΧΙΚΟ
    Γραμμή:   3, Ν. ΦΙΛΑΔΕΛΦΕΙΑ - ΑΝΩ ΠΑΤΗΣΙΑ - ΓΗΡΟΚΟΜΕΙΟ
    Γραμμή: 022, Ν. ΚΥΨΕΛΗ - ΜΑΡΑΣΛΕΙΟΣ
    Γραμμή: 054, ΠΕΡΙΣΣΟΣ - ΑΚΑΔΗΜΙΑ - ΜΕΤΑΜΟΡΦΩΣΗ
    Γραμμή: 060, ΜΟΥΣΕΙΟ - ΑΚΑΔΗΜΙΑ - ΛΥΚΑΒΗΤΤΟΣ (ΚΥΚΛΙΚΗ)
    Γραμμή: 100, ΠΛ. ΚΟΥΜΟΥΝΔΟΥΡΟΥ - ΚΟΛΩΝΑΚΙ - ΑΓΟΡΑ (ΚΥΚΛΙΚΗ)
    Γραμμή: 224, ΚΑΙΣΑΡΙΑΝΗ - ΕΛ. ΒΕΝΙΖΕΛΟΥ
    Γραμμή: 608, ΓΑΛΑΤΣΙ - ΑΚΑΔΗΜΙΑ - ΝΕΚΡ. ΖΩΓΡΑΦΟΥ
    Γραμμή: 622, ΓΟΥΔΗ - ΑΝΩ ΓΑΛΑΤΣΙ
    Γραμμή: 732, ΑΓ. ΦΑΝΟΥΡΙΟΣ - ΑΚΑΔΗΜΙΑ - ΖΩΟΔ. ΠΗΓΗ
    Γραμμή:  Α7, ΣΤΟΥΡΝΑΡΗ - ΚΗΦΙΣΙΑ (ΠΛ.ΠΛΑΤΑΝΟΥ)
```

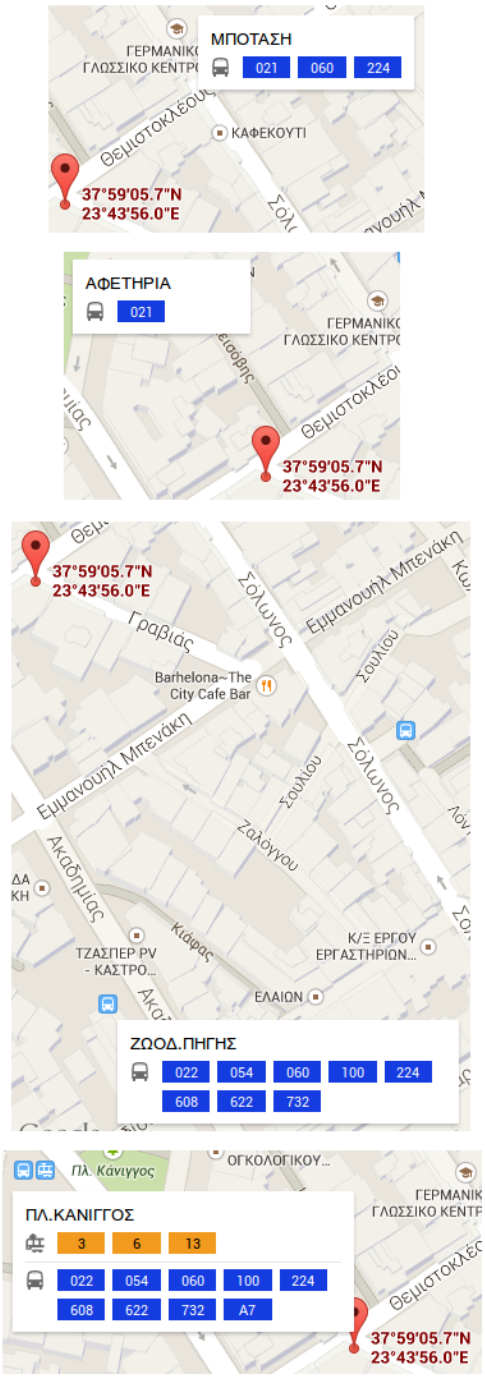
Οι συντεταγμένες του συγκεκριμένου σχολείου όπως έχουν δοθεί στα δεδομένα, είναι 37.984928 γεωγραφικό μήκος και 23.732217 γεωγραφικό πλάτος. Στην εικόνα 1 βλέπουμε και στο χάρτη[1] ότι πράγματι αυτές οι τέσσερις στάσεις βρίσκονται κοντά στο σημείο που αναζητήσαμε και ότι έχουμε βρει σωστά τα δρομολόγια που εξυπηρετούν την κάθε στάση.

Web Service

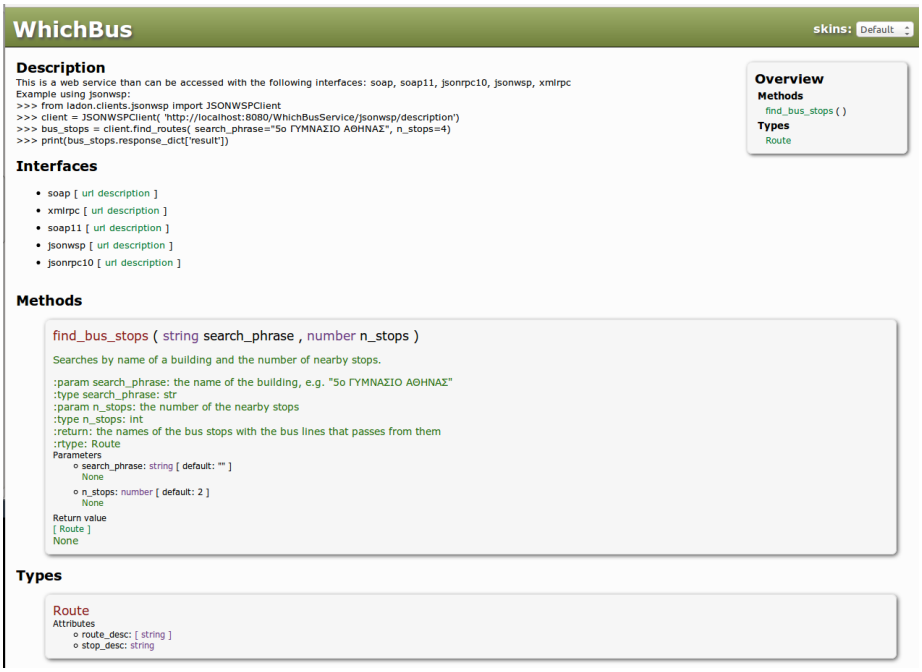
Για την χρήση της εφαρμογής ως Web Service χρειαζόμαστε έναν Server στον οποίο θα μπορούν να γίνονται οι κλήσεις. Για τη συγκεκριμένη εφαρμογή, τόσο για να αποτελέσει Web Service, όσο και για τον Server που θα την φιλοξενήσει, χρησιμοποιήθηκε το Landon Framework (`http://ladonize.org`).

Μαζί με τον κώδικα της εφαρμογής, υπάρχει ένα αρχείο με το όνομα `runserver.py` το οποίο πρέπει να εκτελεστεί. Όσο εκτελείται αυτό το αρχείο, μπορούμε στο url `http://localhost:8080/WhichBus` να δούμε την περιγραφή του Web Service, τα Interfaces που υποστηρίζει, τη μέθοδο που μπορούμε να καλέσουμε και τους τύπους (classes) που χρησιμοποιούνται. Τα στοιχεία που περιγράφονται εδώ φαίνονται και στην εικόνα 2.

[1] Τα δεδομένα είναι από το Google Maps.



Εικόνα 1: Επιβεβαίωση των αποτελεσμάτων αναζήτησης στον χάρτη



Εικόνα 2: Σελίδα που βλέπουμε στο url http://localhost:8080/WhichBus

Για τη κλήση του Web Service μέσω του Interface JSONWSP, πέρα από το παράδειγμα που φαίνεται στη σελίδα `http://localhost:8080/WhichBus` (βλ. εικόνα 2), στο φάκελο της εφαρμογής, υπάρχει ένα αρχείο με το όνομα `test_service.py` το οποίο μπορεί να δει κάποιος ως παράδειγμα ή ακόμη και να εκτελέσει με την εντολή `python3 test_service.py` σε τερματικό. Το παραπάνω αρχείο αναζητεί το 5ο ΓΥΜΝΑΣΙΟ ΑΘΗΝΑΣ και τέσσερις κοντινές στάσεις. Το αποτέλεσμα της εκτέλεσης φαίνεται παρακάτω.

```
[ { 'route_desc': [ '021, ΚΑΝΙΓΓΟΣ - ΓΚΥΖΗ',
                    '060, ΜΟΥΣΕΙΟ - ΑΚΑΔΗΜΙΑ - ΛΥΚΑΒΗΤΤΟΣ (ΚΥΚΛΙΚΗ)',
                    '224, ΚΑΙΣΑΡΙΑΝΗ - ΕΛ. ΒΕΝΙΖΕΛΟΥ'],
    'stop_desc': 'ΜΠΟΤΑΣΗ, Επί της ΣΟΛΩΝΟΣ'},
  { 'route_desc': ['021, ΚΑΝΙΓΓΟΣ - ΓΚΥΖΗ'],
    'stop_desc': 'ΑΦΕΤΗΡΙΑ, Επί της ΤΖΩΡΤΖ'},
  { 'route_desc': [ '022, Ν. ΚΥΨΕΛΗ - ΜΑΡΑΣΛΕΙΟΣ',
                    '054, ΠΕΡΙΣΣΟΣ - ΑΚΑΔΗΜΙΑ - ΜΕΤΑΜΟΡΦΩΣΗ',
                    '060, ΜΟΥΣΕΙΟ - ΑΚΑΔΗΜΙΑ - ΛΥΚΑΒΗΤΤΟΣ (ΚΥΚΛΙΚΗ)',
                  '100, ΠΛ. ΚΟΥΜΟΥΝΔΟΥΡΟΥ - ΚΟΛΩΝΑΚΙ - ΑΓΟΡΑ (ΚΥΚΛΙΚΗ)',
                    '224, ΚΑΙΣΑΡΙΑΝΗ - ΕΛ. ΒΕΝΙΖΕΛΟΥ',
                    '608, ΓΑΛΑΤΣΙ - ΑΚΑΔΗΜΙΑ - ΝΕΚΡ. ΖΩΓΡΑΦΟΥ',
                    '622, ΓΟΥΔΗ - ΑΝΩ ΓΑΛΑΤΣΙ',
                    '732, ΑΓ. ΦΑΝΟΥΡΙΟΣ - ΑΚΑΔΗΜΙΑ - ΖΩΟΔ. ΠΗΓΗ'],
    'stop_desc': 'ΖΩΟΔ.ΠΗΓΗΣ, Επί της ΑΚΑΔΗΜΙΑΣ'},
  { 'route_desc': [ '  6, ΙΠΠΟΚΡΑΤΟΥΣ - Ν. ΦΙΛΑΔΕΛΦΕΙΑ - ΚΟΚ. ΜΥΛΟΣ',
                    ' 13, ΛΑΜΠΡΙΝΗ - ΠΛ. ΚΑΝΙΓΓΟΣ - Ν. ΨΥΧΙΚΟ',
                    '  3, Ν. ΦΙΛΑΔΕΛΦΕΙΑ - ΑΝΩ ΠΑΤΗΣΙΑ - ΓΗΡΟΚΟΜΕΙΟ',
                    '022, Ν. ΚΥΨΕΛΗ - ΜΑΡΑΣΛΕΙΟΣ',
                    '054, ΠΕΡΙΣΣΟΣ - ΑΚΑΔΗΜΙΑ - ΜΕΤΑΜΟΡΦΩΣΗ',
                    '060, ΜΟΥΣΕΙΟ - ΑΚΑΔΗΜΙΑ - ΛΥΚΑΒΗΤΤΟΣ (ΚΥΚΛΙΚΗ)',
                  '100, ΠΛ. ΚΟΥΜΟΥΝΔΟΥΡΟΥ - ΚΟΛΩΝΑΚΙ - ΑΓΟΡΑ (ΚΥΚΛΙΚΗ)',
                    '224, ΚΑΙΣΑΡΙΑΝΗ - ΕΛ. ΒΕΝΙΖΕΛΟΥ',
                    '608, ΓΑΛΑΤΣΙ - ΑΚΑΔΗΜΙΑ - ΝΕΚΡ. ΖΩΓΡΑΦΟΥ',
```

```
                '622, ΓΟΥΔΗ - ΑΝΩ ΓΑΛΑΤΣΙ',
                '732, ΑΓ. ΦΑΝΟΥΡΙΟΣ - ΑΚΑΔΗΜΙΑ - ΖΩΟΔ. ΠΗΓΗ',
                ' Α7, ΣΤΟΥΡΝΑΡΗ - ΚΗΦΙΣΙΑ (ΠΛ.ΠΛΑΤΑΝΟΥ)'],
 'stop_desc': 'ΠΛ.ΚΑΝΙΓΓΟΣ, Επί της ΠΛ.ΚΑΝΙΓΓΟΣ'}]
```

Βλέπουμε ότι τα αποτελέσματα είναι τα ίδια με το παράδειγμα εκτέλεσης ως αυτόνομης εφαρμογής, κάτι το οποίο είναι φυσικά αναμενόμενο.

Παρατηρήσεις

Ακολουθούν μερικές σημαντικές παρατηρήσεις για τη λειτουργία της εφαρμογής.

1. Αν στο παραπάνω παράδειγμα αντί για `5ο ΓΥΜΝΑΣΙΟ ΑΘΗΝΑΣ` ζητούσαμε `5ο ΓΥΜΝΑΣΙΟ ΑΘΗΝΩΝ`, η εφαρμογή θα μας ζητούσε ξανά την ονομασία του κτιρίου.
2. Αρκετά κτίρια έχουν αλλάξει ονομασία χωρίς αυτό να αποτυπώνεται στα διαθέσιμα δεδομένα. Για παράδειγμα, τα πρώην Ενιαία Λύκεια ονομάζονται τώρα Γενικά Λύκεια. Στα δεδομένα αναφέρονται ως Ενιαία.
3. Τα ονόματα των κτιρίων είναι γραμμένα με κεφαλαία γράμματα και θα πρέπει στο πεδίο αναζήτησης ο χρήστης να χρησιμοποιήσει κεφαλαία.
4. Οι συντεταγμένες στα δεδομένα δεν ανταποκρίνονται στις ακριβείς τοποθεσίες των κτιρίων. Π.χ. στο παράδειγμα του 5ου Γυμνασίου Αθηνών οι πραγματικές συντεταγμένες είναι 37.98623, 23.73595 και όχι 37.984928, 23.732217 που έχουμε στα δεδομένα το οποίο είναι μια διαφορά περίπου 300 μέτρων σε ευθεία γραμμή.
5. Κάποια δρομολόγια έχουν αλλάξει διαδρομή ή έχουν καταργηθεί ή έχουν δημιουργηθεί από τον Δεκέμβριο του 2013 όπου διαθέτουμε τα δεδομένα του ΟΑΣΑ.

Για την επίλυση των παραπάνω θεμάτων μπορούν να γίνουν προσεγγίσεις όπως.

1. Χρήση της ρίζας των λέξεων τόσο στους όρους αναζήτησης όσο και στα δεδομένα. Έτσι οι όροι ΑΘΗΝΑΣ και ΑΘΗΝΩΝ γίνονται ΑΘΗΝ και στην αναζήτηση θα ταιριάξουν. Για να γίνει αυτό χρειαζόμαστε έναν stemmer για την Ελληνική γλώσσα.
2. Με μία λίστα συνώνυμων όρων όπου το ΕΝΙΑΙΟ και το ΓΕΝΙΚΟ να σημαίνουν το ίδιο πράγμα για την εφαρμογή. Το μειονέκτημα αυτής της μεθόδου είναι ότι πρέπει να συντηρείται από τον προγραμματιστή της εφαρμογής.
3. Με μετατροπή των πεζών σε κεφαλαία και αντικατάσταση των τονισμένων φωνηέντων με άτονα.
4. Για αυτό το πρόβλημα, όπως και για το 5ο, χρειαζόμαστε πιο πρόσφατα δεδομένα και με μεγαλύτερη ακρίβεια στις συντεταγμένες.

Κώδικας Εφαρμογής

Η εφαρμογή υλοποιήθηκε χρησιμοποιώντας τη γλώσσα προγραμματισμού `Python` και συγκεκριμένα την έκδοση `3.4`. Για την επιτυχή εκτέλεση χρειάζονται τα πακέτα `pandas`, `sklearn` και `landon`. Επιπλέον, για την εκτέλεση του `test_service.py` ή γενικά για χρήση του Web Service από Pyhton κώδικα, χρειάζεται η βιβλιοθήκη `suds- jurko`.

Η λογική που ακολουθείται είναι η εξής: Από τα δεδομένα των στάσεων, δημιουργούμε έναν πίνακα που έχει μόνο τις συνταγμένες (αρχείο stops.txt). Ο πίνακας αυτός χρησιμοποιείται από τον αλγόριθμο NearestNeighbors. Συγκεκριμένα, δίνοντας ένα κτίριο στην αναζήτηση, απομονώνουμε τις συντεταγμένες του (αρχείο dhmosia_ktiria_attikis.csv). Οι συντεταγμένες δίνονται στον αλγόριθμο NearestNeighbors και εκείνος μας επιστρέφει τα σημεία (που αντιστοιχούν σε στάσεις) και είναι πιο κοντά στις συντεταγμένες της αναζήτησης. Επιστρέφονται τόσα σημεία όσα έχει ορίσει ο χρήστης.

Τα αποτελέσματα της παραπάνω διαδικασίας είναι μια λίστα με κωδικούς στάσεων (stop_id). Για κάθε stop_id υπάρχει ένα σύνολο από δρομολόγια που το εξυπηρετούν και αφορούν την ίδια λεωφορειακή γραμμή (αρχείο stop_times.txt). Αυτό γίνεται γιατί τα δρομολόγια αντιστοιχούν σε διαφορετικές μέρες και ώρες της εβδομάδας. Κάθε δρομολόγιο, όμως, στην περιγραφή του έχει τον αριθμό της γραμμής (route_id) τον οποίο απομονώνουμε και κρατάμε. Τέλος, με το route_id βρίσκουμε την ονομασία της γραμμής (αρχείο routes.txt).

Ακολουθεί ο κώδικας της εφαρμογής (αρχείο `which_bus.py`.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
# which_bus.py
#
# Copyright 2014 Aris Fergadis <fergadis.aris@gmail.com>
#
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 of the License, or
#   (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#   along with this program; if not, write to the Free Software
#   Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
#   MA 02110-1301, USA.
#
#
from ladon.ladonizer import ladonize
from ladon.types.ladontype import LadonType
from ladon.compat import PORTABLE_STRING
from sklearn.neighbors import NearestNeighbors
from pandas import read_csv, read_table, DataFrame


class Route(LadonType):
    """
    Stores the stop description and a list with
    the bus lines that pass from this stop (routes).
    """
    stop_desc = PORTABLE_STRING
    route_desc = [PORTABLE_STRING]


class WhichBus(object):
    """
    This is a web service than can be accessed with the following
    interfaces: soap, soap11, jsonrpc10, jsonwsp, xmlrpc</br>
    Example using jsonwsp:</br>
    >>> from ladon.clients.jsonwsp import JSONWSPClient</br>
    >>> client = JSONWSPClient(\
   'http://localhost:8080/WhichBusService/jsonwsp/description')</br>
```

```
 >>> bus_stops = client.find_routes(\
 search_phrase="5o ΓΥΜΝΑΣΙΟ ΑΘΗΝΑΣ", n_stops=4)</br>
 >>> print(bus_stops.response_dict['result'])</br>
 """

 def __init__(self):
     # Load data
     self.public_sector = read_table(
         "data/dhmosia_kthria/dhmosia_kthria_attikis.csv",
         index_col="ktirio_ypiresia", sep=';')
     self.stops = read_csv("data/oasa/stops.txt")
     self.routes = read_csv("data/oasa/routes.txt",
                            index_col="route_short_name")
     self.stop_times = read_csv("data/oasa/stop_times.txt",
                                index_col="stop_id")
     # Get only the coordinates of the bus stops to make the
     # "training" data
     self.coordinates = DataFrame.as_matrix(
         self.stops, columns=["stop_lon", "stop_lat"])
     self.nbrs = NearestNeighbors().fit(self.coordinates)

 @ladonize(PORTABLE_STRING, int, rtype=[Route])
def find_bus_stops(self, search_phrase=PORTABLE_STRING(''), n_stops=2):
     """
   <p>Searches by name of a building and the number of nearby stops.</p>

     :param search_phrase: the name of the building,
            e.g. "5o ΓΥΜΝΑΣΙΟ ΑΘΗΝΑΣ"</br>
     :type search_phrase: str</br>
     :param n_stops: the number of the nearby stops</br>
     :type n_stops: int</br>
     :return: the names of the bus stops with the
              bus lines that passes from them</br>
     :rtype: Route</br>
     """
     routes_results = []
     lon = self.public_sector.ix[search_phrase].longitude
     # lon has 23,7536 but we want 23.7536
     lon = lon.replace(',', '.')
     lat = self.public_sector.ix[search_phrase].latitude
     lat = lat.replace(',', '.')
     search_point = [lon.strip(), lat.strip()]
     results = self.nbrs.kneighbors(search_point, n_stops,
                                    return_distance=False)
     for r in range(n_stops):
         route_for_stop = Route()
         route_for_stop.route_desc = []
         # Find the stop id from stops
         stop_id = self.stops.ix[results[0, r]]["stop_id"]
         route_for_stop.stop_desc = "{}, {}". \
             format(self.stops.ix[results[0, r]].stop_name,
                    self.stops.ix[results[0, r]].stop_desc)
         # Find the trips ids for that stop
         trips_of_stop_id = self.stop_times.ix[stop_id]
```

```
            route_ids = []  # List with all routes from the stop
            for trip in trips_of_stop_id.trip_id:
              # trip has the form 7038869-ΤΗΛΕΜΑ-Τ5-Παρασκευή-03
                # We want the 3rd item (T5)
                route_num = trip.split('-')[2]
                # We get many trip ids which are for the same
                # route (T5) but for different days and times
                if not route_num in route_ids:
                    # We keep the route id (T5) only one time,
                    # and the rest route ids
                    route_ids.append(route_num)

            for route_id in route_ids:
                route_for_stop.route_desc +=\
                    ["{:>3s}, {}".format(route_id,
                            self.routes.ix[route_id].route_long_name)]

            routes_results += [route_for_stop]

        return routes_results

if __name__ == "__main__":
    search = input(
      "Δώστε την ονομασία του κτιρίου (π.χ. 5ο ΓΥΜΝΑΣΙΟ ΑΘΗΝΑΣ): ")

    while True:
        try:
        n_stops = int(input("Πλήθος στάσεων προς αναζήτηση (π.χ. 3): "))
            if n_stops <= 0:
           print("Παρακαλώ δώστε αριθμό μεγαλύτερο από το μηδέν.")
            else:
                break  # We've got the number of stops
        except ValueError:
            print("Παρακαλώ δώστε αριθμό.")

    wb = WhichBus()
    find_results = wb.find_bus_stops(search, n_stops)
    for i in range(len(find_results)):
     print("Ονομασία Στάσης: {}".format(find_results[i].stop_desc))
        for route in find_results[i].route_desc:
            print("\tΓραμμή: {}".format(route))
        print()
```

Ο κώδικας για τον Server , αρχείο `runserver.py`, είναι ο εξής.

```
# -*- coding: utf-8 -*-
import wsgiref.simple_server
from os.path import abspath, dirname, join

from ladon.server.wsgi import LadonWSGIApplication


scriptdir = dirname(abspath(__file__))
service_modules = ['which_bus']
```

```
# Create the WSGI Application
application = LadonWSGIApplication(
    service_modules,
    [scriptdir, join(scriptdir, 'appearance')],
    catalog_name='Which Bus Web Service',
    catalog_desc='Click on the link to see how'
                 'to use the WhichBus service')

if __name__ == '__main__':
    # Starting the server from command-line will
    # create a stand-alone server on port 8080
    port = 8080
    print("\nExample services are running on port %(port)s."
          "\nView browsable API at http://localhost:%(port)s\n"
          % {'port': port})

    server = wsgiref.simple_server.make_server(
        '', port, application)
    server.serve_forever()
```

Τέλος, παρατίθεται και ο κώδικας που καλεί το Web Service και μπορεί να αποτελέσει ένα παράδειγμα για χρήση από άλλους προγραμματιστές (αρχείο `test_service.py`).

```
# -*- coding: utf-8 -*-
"""
Tests the which_bus service. The server should be started.
If not, please run runserver.py first.
"""
from ladon.clients.jsonwsp import JSONWSPClient
import pprint


def print_result(jsonwsp_resp):
    if jsonwsp_resp.status == 200:
     pprint.pprint(jsonwsp_resp.response_dict['result'], indent=2)
    else:
     print("A problem occurred while communicating with the service:\n")
        print(jsonwsp_resp.response_body)

client = JSONWSPClient('http://localhost:8080/WhichBus/jsonwsp/description')
bus_stops = client.find_bus_stops(search_phrase="5ο ΓΥΜΝΑΣΙΟ ΑΘΗΝΑΣ", n_stops=4)
print_result(bus_stops)
```

Για τον κώδικα μαζί με τα δεδομένα κάντε κλικ στο παρακάτω url:
`https://copy.com/Hz0usesLwLnT`

Κατακλείδα

Με απλό συνδυασμό ανοιχτών δεδομένων και με λίγες γραμμές κώδικα, δημιουργήσαμε μία εφαρμογή η οποία μπορεί να είναι χρήσιμη στα άτομα εκείνα που κινούνται τακτικά με τα Μέσα Μαζικής Μεταφοράς.

Ως επέκταση θα μπορούσαμε μαζί με την περιγραφή της στάσης να επιστρέφουμε και τις συντεταγμένες της. Έτσι, θα μπορούσε κάποιος να χρησιμοποιήσει τις πληροφορίες αυτές και να

τις εμφανίσει σε ένα χάρτη σε μια ιστοσελίδα ή ακόμη και σε μια εφαρμογή για κινητό τηλέφωνο. Επίσης, χρήσιμη θα ήταν η δυνατότητα οι χρήστες να μπορούν να διορθώνουν στοιχεία όπως, ονομασίες κτιρίων, συντεταγμένες, στοιχεία δρομολογίων ή ακόμη και να προσθέτουν νέα.