DATA MARKET

AUSTRIA

www.datamarket.at

Mobility Pilot - Service Description

Deliverable number	*D8.3*
Dissemination level	*Private*
Delivery date	*30.09.2018*
Status	*Version 1.1*
Author(s)	*Bernhard Niedermayer (Catalysts), Lukas Winkler (Catalysts)*

The Data Market Austria Project has received funding from the programme "ICT of the Future" of the Austrian Research Promotion Agency (FFG) and the Austrian Ministry for Transport, Innovation and Technology (Project 855404)

Executive Summary

This document briefly describes how to build and run the Taxi Demand Heatmap service. A detailed description of the service itself can be found in the deliverable D8.2.

The main part of the deliverable D8.3 consists of the source code of the first running prototype. This document accompanies the source code and describes the pilot components and usage.

Table of Contents

1. Introduction

The current version of the Heatmap service is a prototype, generating a map showing areas of increased demand for taxis. Its general purpose is to combine several data sources to predict the demand for taxi rides in the Vienna area. The computation of this demand is based upon a model, which is trained in a preceding data analysis step.

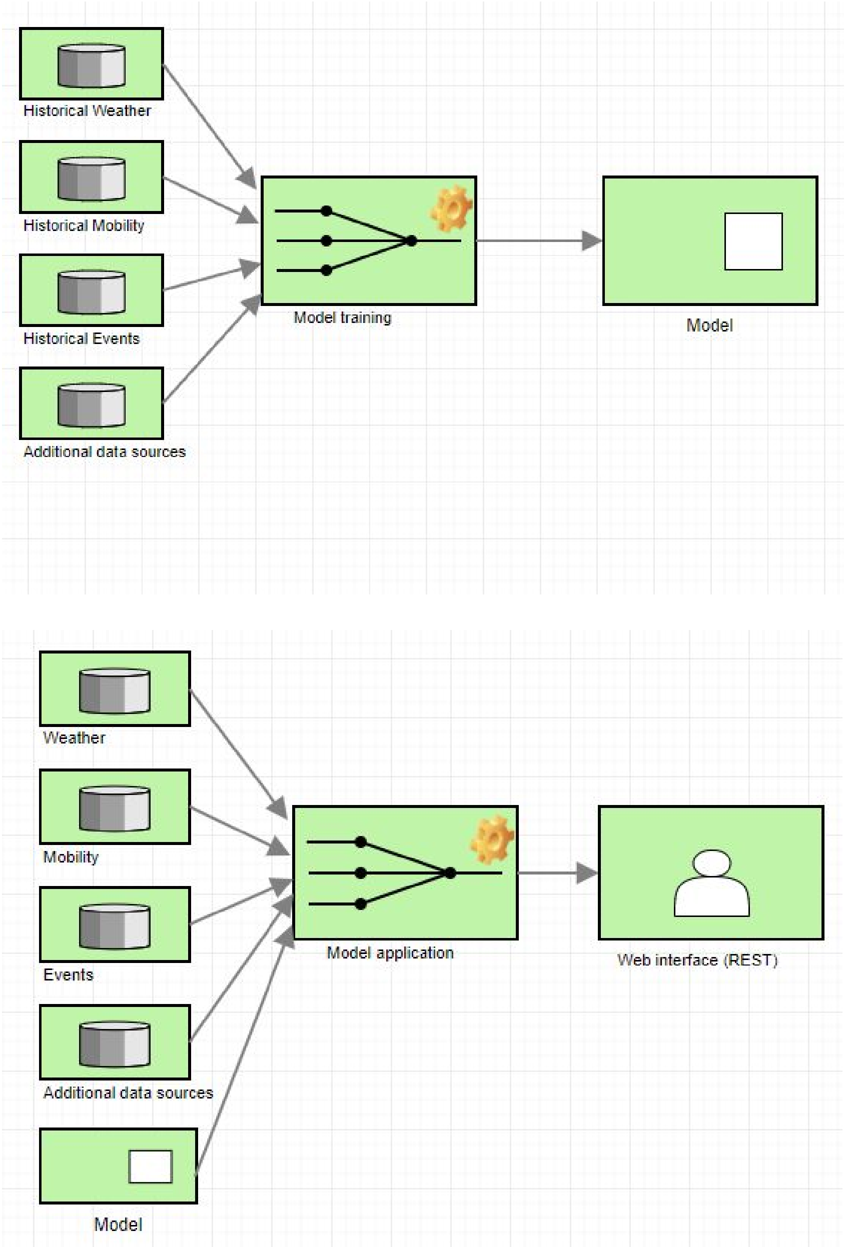


Figure 1: Model training and application (prediction)

The data analysis is done using the Python stack (including libraries such as numpy[1], matplotlib[2], scikit learn[3], pandas[4], etc.). Geo-referencing is done using gdal[5].

[1] http://www.numpy.org
[2] https://matplotlib.org
[3] http://scikit-learn.org

According to this, the prediction component is realized using the Python stack as well. For the implementation of the web service Flask[6] has been used.

Figure 2 shows an exemplary visualization of the heatmap as an OpenStreetMap overlay and the possibility to include and exclude different influencing factors into/from the prediction.

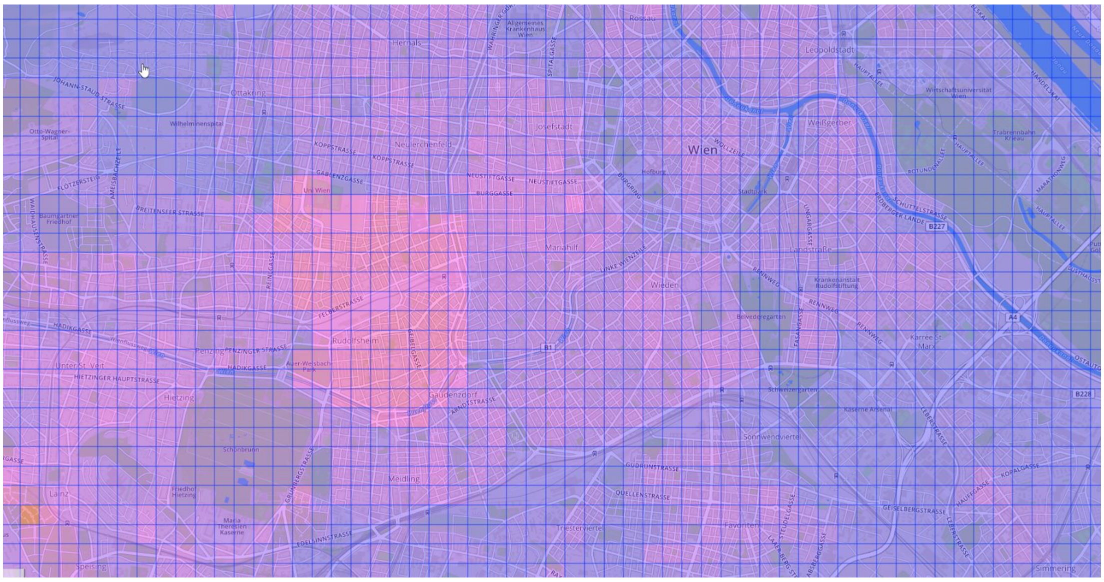


Figure 2 - Taxi demand heatmap of the Vienna area (2018-11-02 12:45)

Throughout the development and continuous improvement of the taxi demand heatmap we allow to show event markers on the overlay for better traceability of computed results. Figure 3 and 4 show a detail of the same heatmap as in Figures 2 including these markers.

[4] https://pandas.pydata.org
[5] https://www.gdal.org
[6] http://flask.pocoo.org

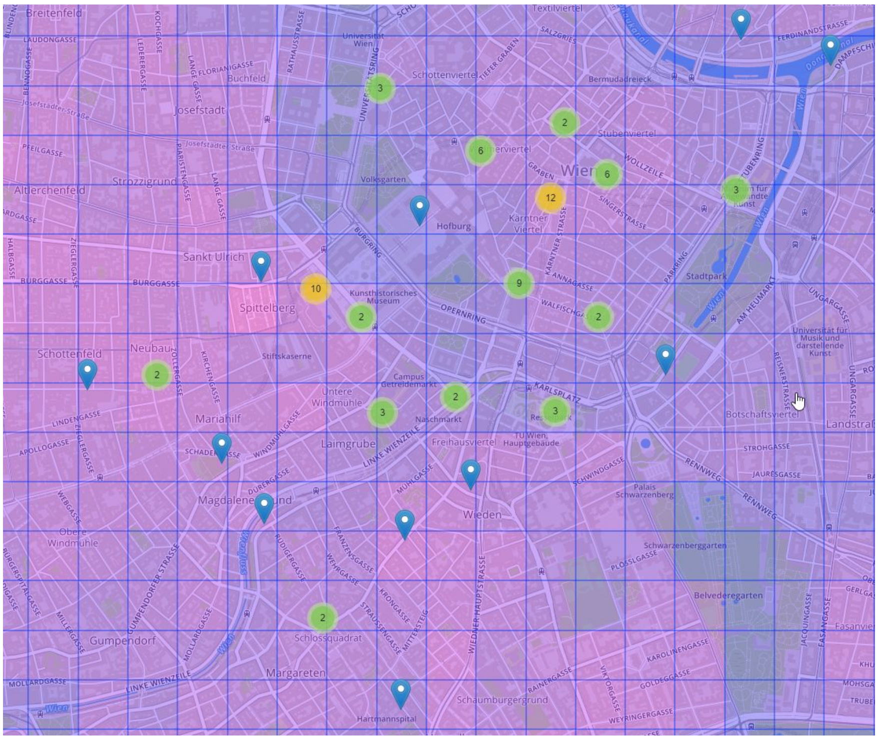


Figure 3 - Taxi demand heatmap with event markers

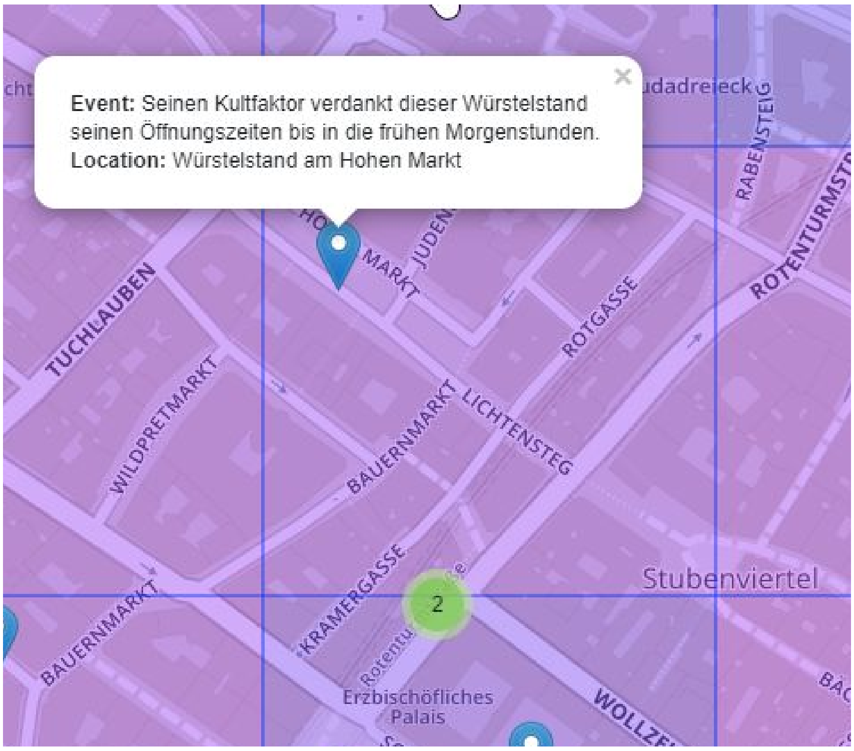
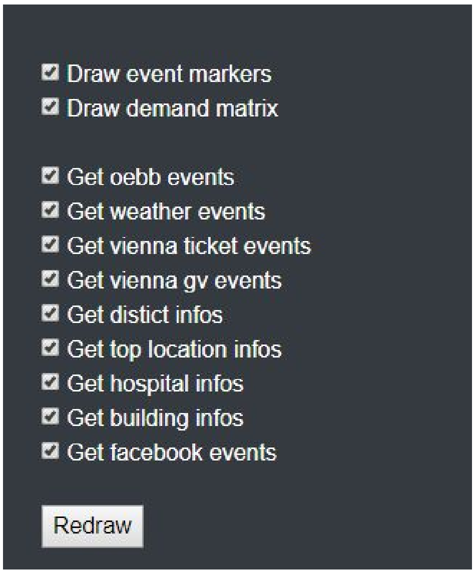

Event: Seinen Kultfaktor verdankt dieser Würstelstand seinen Öffnungszeiten bis in die frühen Morgenstunden.
Location: Würstelstand am Hohen Markt

Figure 4 - Taxi demand heatmap with event markers and event info

Figure 5 depicts an excerpt of the development stage GUI allowing to easily include or exclude individual input data sources.



☑ Draw event markers
☑ Draw demand matrix

☑ Get oebb events
☑ Get weather events
☑ Get vienna ticket events
☑ Get vienna gv events
☑ Get distict infos
☑ Get top location infos
☑ Get hospital infos
☑ Get building infos
☑ Get facebook events

Figure 5 - Configuration menu of the taxi demand heatmap allowing to include different sets of input data

2. Technical Description

These estimations are based upon several data sources, listed below:

- Train delay information from ÖBB at http://zugradar.oebb.at
- Local event info from various sources (Facebook, https://www.wien.gv.at/vadb/internet/, etc.)
- Weather information
- Districts' population numbers
- Points of interests (event locations, restaurants, bars, hospitals, etc.)

Components

The main components of the source code are:

heatmap1.html

Opening this file in the browser will query a web service (implemented in server.py) for current demand estimates and display them on a map.

server.py

This is the service serving the demand info as a geojson. Because it takes a few seconds to generate the demand estimations (because other services are being queried), a new geojson is not generated on each request, but only after a certain amount of time has passed.

demand.py

Main file for generating taxi demand estimations from the different data sources.

fb_events.py

Queries the Facebook Graph API for events in Vienna. The API does not offer an option to directly query for events by geographic location, so we have to work around that. In addition, querying the Facebook graph API requires authentication and permissions.

- **Search for events by keyword:** there is the option to search for events containing a search term in their title. We therefore search for events containing "Vienna" and "Wien".
- **Search for events by venue:** The Facebookgraph API allows to search for event venues near a position given by latitude and longitude. This only works well for venues which are at mosta few hundred meters away from the query location. We therefore construct a rectangular grid over Vienna and search for event venues within each grid cell separately. Since these locations only vary slowly, this search only needs to be done sporadically. In the next step we go through all of the returned event venues and query for events at those venues. Since there are tens of thousands of defined venues within Vienna (most of them very small and unimportant) and the graph API only allows up to 50 venues per query, we have to send a lot of queries. In order to not get rate limited by Facebook, we send only one query per second. The whole procedure therefore takes around 10-15 minutes. But since the events do not change much within a single day, we only do these mass queries once per day.

- **Authorization for the Facebook Graph API**

  There are two different kinds of authorization requests to query the Facebook graph API: as an application or as a user. Both require to send a key along with the query, but users have more access rights than applications. In our context, this is relevant for the "search for events by keyword"-feature. This kind of search can only be done by users. Access keys for applications do not change, whereas user keys are only valid for less than an hour and there needs to be a user interaction to get new credentials. Therefore, the searching for events by locations can be done with the app credentials which are hardcoded in the application. For searching for events by keywords, it is necessary to generate a user key at https://developers.facebook.com/tools/explorer and pass it to the application as a command line parameter.

Usage

The heatmap service is initialized and started as follows:

Remark: The app has been registered under one of the pilot's developers personal Facebook account. In order to run and test it, the app need to be officially submitted to and reviewed by Facebook under an operator’s account in order to get access to the Facebook API.

1. Generate List of Locations from Facebook: (do this once or rarely)

   ```
   python fb_events.py get_locations
   ```

   This takes about 25 minutes to run.

2. Generate List of Events from Facebook: (do this daily)

   ```
   python fb_events.py get_events --key=[user-key]
   ```

   where the user-key is generated at https://developers.facebook.com/tools/explorer]

   alternatively get events by searching by venue only (app key is sufficient for this query):

   ```
   python fb_events.py get_events
   ```

   This takes about 10 minutes to run.

3. Start server:

   ```
   python server.py --window_size=1000
   ```

   The size of the time window is set very high to make sure that we definitely have events to display. When actually using the map as a decision tool, recommended values are 15, 30 or 60 minutes.

4. Open heatmap1.html with a web browser