Data Science Project Report– Junior Team

Content

Introduction and Task Description

The purpose of this report is to show the progress of the junior team during the first half of the Data-Science Project and to document our work in order for new teams to be able to read into the topic quickly and to reproduce our solutions.

Our tasks can be separated into the following fields:

1. Virtual infrastructure:
    - Setup a virtual machine
    - Install and configure Pelias and Elasticsearch
    - Install and test routing engines
2. Data acquisition and preparation:
    - Gather postcode data of Europe from different sources
    - Merge the postcode data into a single custom source
    - Calculate 2-digit postcode centroids from the custom dataset
3. Geocoding and routing:
    - Test geocoding with Pelias based on the prepared 2-digit postcode centroids
    - Test routing between 2-digit centroids with a routing engine (Valhalla or Graphhopper)

The Task of the junior team was to test Pelias as a geocoding service and an alternative to Nominatim. Therefore, it was necessary to build up a database of postcodes in Europe and to calculate 2-digit postcode centroids as well as build a complete map of Europe based on Openstreetmaps, Whosonfirst, Geonames and Postcodeinfo data. Furthermore, it was our task to find alternatives for the routing engine Graphhopper.

In Addition to those tasks it was required to document each single step of our work properly to be able to reproduce everything at a later point in time.

Pelias

General Info

Capabilities

- Geocoding is the process of taking input text, such as an address or the name of a place and returning a latitude/longitude location on the Earth's surface for that place.
- Reverse geocoding is the opposite: returning a list of places near a given latitude/longitude point.
- Completely open-source and MIT licensed
- A powerful data import architecture: Pelias supports many open-data projects out of the box but also works great with private data
- Support for searching and displaying results in many languages
- Fast and accurate autocomplete for user-facing geocoding
- Support for many result types: addresses, venues, cities, countries, and more
- Modular design, so you don't need to be an expert in everything to make changes
- Easy installation with minimal external dependencies

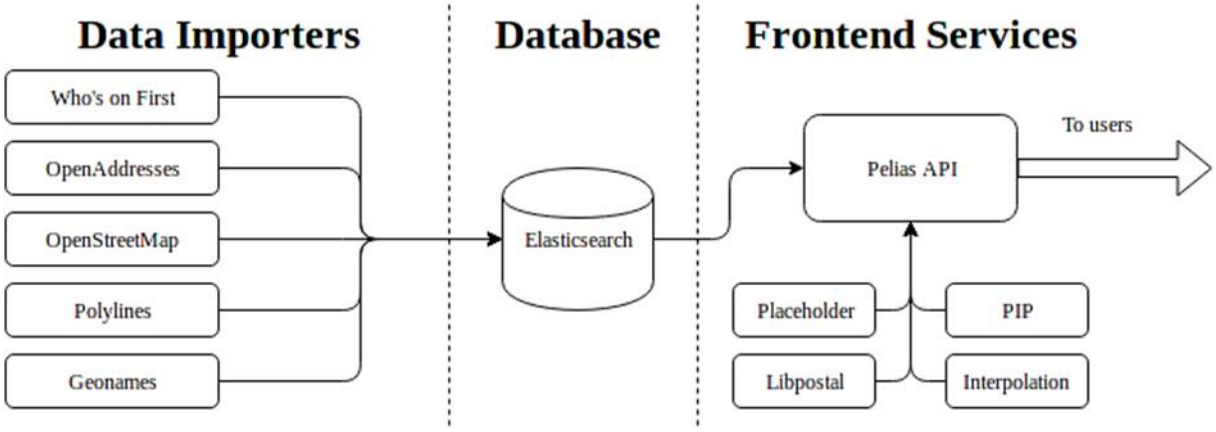

Figure 1: The architecture of the pelias service [1]

Data Importers

The importers filter, normalize, and ingest geographic datasets into the Pelias database. Currently there are five officially supported importers:

- **OpenStreetMap**: supports importing nodes and ways from OpenStreetMap
- **OpenAddresses**: supports importing the hundreds of millions of global addresses collected from various authoritative government sources by OpenAddresses
- **Who's on First**: supports importing admin areas and venues from Who's on First
- **Geonames**: supports importing admin records and venues from Geonames
- **Polylines**: supports any data in the Google Polyline format. It's mainly used to import roads from OpenStreetMap
- **Custom Data Importer:** creates a Pelias record for each row in a CSV file. Each row must define a source, latitude, longitude, and either an address, name, or both. This feature was used to import two-digit postcodes into Pelias.

Database

The underlying datastore that powers the search results and does query-lifting is Elasticsearch. Currently version 2.4 is supported, with plans to support 5.x soon. The developers built a tool called pelias-schema that sets up Elasticsearch indices properly for Pelias.

Frontend Services

This is where the actual geocoding process happens and includes the components that users interact with when performing geocoding queries. These services are:

- **API**: The API service defines the Pelias API and talks to Elasticsearch or other services as needed to perform queries.
- **Placeholder**: A service built specifically to capture the relationship between administrative areas (a catch-all term meaning anything like a city, state, country, etc). Elasticsearch does not handle relational data very well, so Placeholder was built specifically to manage this piece.
- **PIP**: For reverse geocoding, it's important to be able to perform point-in-polygon(PIP) calculations quickly. The PIP service is the only component of Pelias that actually understands polygon geometries, and it is very good at quickly determining which admin area polygons a given point lies in.
- **Libpostal**: Pelias uses the libpostal project for parsing addresses using the power of machine learning. A Go service built by the Who's on First team makes this happen quickly and efficiently.
- **Interpolation**: This service knows all about addresses and streets. With that knowledge, it is able to supplement the *known* addresses that are stored directly in Elasticsearch and return fairly accurate *estimated* address results for many more queries than would otherwise be not be possible. Interpolated address data is based on openstreetmaps data. Calculating and importing interpolated data took around 50 hours.

Dependencies

These are software projects that are not used directly but are used by other components of Pelias:

- **model:** provide a single library for creating documents that fit the Pelias Elasticsearch schema. This is a core component of Pelias' flexible importer architecture
- **wof-admin-lookup:** A library for performing administrative lookup using point-in-polygon math. Previously included in each of the importers but now only used by the PIP service.
- **query:** This is where most of Elasticsearch's query generation happens.
- **config:** Pelias is very configurable, and all of it is driven from a single JSON file which is called pelias.json. This package provides a library for reading, validating, and working with this configuration. It is used by almost every other Pelias component
- **dbclient:** A Node.js stream library for quickly and efficiently importing records into Elasticsearch

System Requirements

Software Requirements

Node.js:

- Most Pelias code is written in Node.js.
- Version 8 or newer is required, version 10 is recommended for improved performance.

Elasticsearch:

- Version 2.4 or 5.6

SQLite:

- Version 3.11 or newer

Libpostal:

- Pelias relies heavily on the Libpostal address parser. Libpostal requires about 4GB of disk space to download all the required data.

Hardware Requirements

- At a minimum 50GB disk space to download, extract, and process data

- 8GB RAM for a local build, 16GB+ for a full planet build. Pelias needs a little RAM for Elasticsearch, but much more for storing administrative data during import
- As many CPUs as possible. There's no minimum, but Pelias builds are highly parallelizable, so more CPUs will help make it faster.

Actual system used for project (Europe build):

- 1 virtual machine (Ubuntu Linux) with 64 GB RAM, 500GB HDD, 4 CPU cores
- RAM utilization is at ~30 GB, however during the import of openstreetmaps data and calculating polylines from it up to 40 GB of RAM were used. Imports and calculations maxed out all CPU cores. It is possible to reduce the required amount of RAM for imports and calculations. However, this requires splitting up openstreetmap files in smaller files with other tools beforehand.
- Including “raw data” (before the import and calculations) around 400GB of data are persisted on HDD, Elasticsearch uses ~100GB.

Installation and Setup

Pelias can be installed as Docker Image, manually from scratch or with Kubernetes. For testing purposes installing Pelias as a Docker Images is strongly recommended by the developers. Pelias can also be installed manually from scratch, but due to the large amount of dependencies this is not recommended by the developers. To use Pelias in production, the development team suggests an installation with Kubernetes, which is by far the most well tested way to install Pelias according to the development team.

Installation with Docker

On the above mentioned virtual machine, Pelias was installed and maintained witch docker and docker-compose.

Docker:

```
$ sudo apt-get update
$ sudo apt-get install \
apt-transport-https \
   ca-certificates \
curl \
gnupg-agent \
software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ sudo add-apt-repository \   "deb [arch=amd64]
https://download.docker.com/linux/ubuntu \   $(lsb_release -cs) \   stable"
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
$ sudo groupadd docker
$ sudo usermod -aG docker $USER
$ sudo systemctl enable docker
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose
```

Pelias:

Pelias’ git repository provides example projects (e.g. Belgium, Portland-Metro, etc.). Pelias’ “planet”project was used as a starting point for a Europe build. For this Pelias was forked on Github and cloned onto the VM:

```
$ pwd
/home/dataproject/git/pelias-docker/projects/Europe
```

In order to build and run Pelias with data for Europe four configuration files in this folder are needed:

.env:

```
COMPOSE_PROJECT_NAME=pelias
DATA_DIR=/data/pelias-docker-compose/
DOCKER_USER=1003
ENABLE_GEONAMES=true
DATA_DIR: directory where Pelias will store downloaded data. Also used to build
its other services.
DOCKER_USER = All Pelias processes in containers run as non-root users. This user
ID will be used for accessing files on the host filesystem in DATA_DIR.
```

Elasticsearch.yml:

```
network.host: 0.0.0.0
bootstrap.memory_lock: true
indices.breaker.fielddata.limit: 85%
indices.fielddata.cache.size: 75%
thread_pool.bulk.queue_size: 500
thread_pool.index.queue_size: 1000
```

Both thread pool sizes had to be increased since the default values were too small. Pelias importers delivered too much data concurrently for Elasticsearch which resulted in corrupted data.

pelias.json:

In this file all Pelias services are configured:

```
{
  "logger": {
    "level": "info",
    "timestamp": false
  },
  "esclient": {
    "apiVersion": "2.4",
    "hosts": [
      {
        "host": "elasticsearch"
      }
    ]
  },
  "elasticsearch": {
    "settings": {
      "index": {
        "refresh_interval": "10s",
        "number_of_replicas": "0",
        "number_of_shards": "5"
      }
    }
  },
  "acceptance-tests": {
    "endpoints": {
      "docker": "http://api:4000/v1/"
    }
  },
  "api": {
    "targets": {
      "canonical_sources": [
        "whosonfirst",
        "openstreetmap",
        "openaddresses",
        "geonames",
```

```
    "geonamesandpostcodeinfo"
  ],
  "layers_by_source": {
    "geonames2d": [
      "country",
      "postalcode",
      "source",
      "layer"
    ],
    "geonamesandpostcodeinfo": [
      "name",
      "source",
      "layer",
      "lat",
      "lon",
      "country",
      "postalcode"
    ],
    "openstreetmap": [
      "address",
      "venue",
      "street"
    ],
    "openaddresses": [
      "address"
    ],
    "geonames": [
      "country",
      "macroregion",
      "region",
      "county",
      "localadmin",
      "locality",
      "borough",
      "neighbourhood",
      "venue",
      "postalcode"
    ],
    "whosonfirst": [
      "continent",
      "empire",
      "country",
      "dependency",
      "macroregion",
      "region",
      "locality",
      "localadmin",
      "macrocounty",
      "county",
      "macrohood",
      "borough",
      "neighbourhood",
      "microhood",
      "disputed",
      "venue",
      "postalcode",
      "continent",
      "ocean",
      "marinearea"
    ]
```

```
    },
    "source_aliases": {
      "osm": [
        "openstreetmap"
      ],
      "oa": [
        "openaddresses"
      ],
      "gn": [
        "geonames"
      ],
      "wof": [
        "whosonfirst"
      ],
      "2dpg": [
        "geonames2d"
      ]
    }
  },
  "textAnalyzer": "libpostal",
  "services": {
    "pip": {
      "url": "http://pip:4200"
    },
    "libpostal": {
      "url": "http://libpostal:4400"
    },
    "placeholder": {
      "url": "http://placeholder:4100"
    },
    "interpolation": {
      "url": "http://interpolation:4300"
    }
  },
  "defaultParameters": {
    "focus.point.lat": 48.88,
    "focus.point.lon": 2.32
  }
},
"imports": {
  "adminLookup": {
    "enabled": true
  },
  "geonames": {
    "datapath": "/data/geonames",
    "countryCode": "ALL",
    "sourceURL": "http://download.geonames.org/export/dump/"
  },
  "openstreetmap": {
    "download": [
      {
        "sourceURL": "https://download.geofabrik.de/europe-latest.osm.pbf"
      }
    ],
    "leveldbpath": "/tmp",
    "datapath": "/data/openstreetmap",
    "import": [
      {
        "filename": "europe-latest.osm.pbf"
      }
```

```
      ]
    },
    "openaddresses": {
      "datapath": "/data/openaddresses",
      "files": [
      ]
    },
    "polyline": {
      "datapath": "/data/polylines",
      "files": [
        "europe.polyline"
      ]
    },
    "whosonfirst": {
      "datapath": "/data/whosonfirst",
      "sqlite": true,
      "importVenues": false,
      "importPostalcodes": true
    },
    "csv": {
      "datapath": "/data/geonamesandpostcodeinfo/",
      "files": [],
      "download": []
    }
  }
}
```

These services all run as docker containers. Therefore, it is not necessary to provide complete full paths on the host filesystem or IP/DNS addresses. Paths are mapped to the path provided in the docker compose file (further below) and .env. Docker has its own networking and DNS. Services in a docker network can be addressed by using container names and ports can be mapped to host ports.

docker-compose.yml:

```
version: '3'
networks:
  default:
    driver: bridge
services:
  libpostal:
    image: pelias/libpostal-service
    container_name: pelias_libpostal
    user: "${DOCKER_USER}"
    restart: always
    ports: [ "4400:4400" ]
  schema:
    image: pelias/schema:master
    container_name: pelias_schema
    user: "${DOCKER_USER}"
    volumes:
      - "./pelias.json:/code/pelias.json"
  api:
    image: pelias/api:master
    container_name: pelias_api
    user: "${DOCKER_USER}"
    restart: always
    environment: [ "PORT=4000" ]
    ports: [ "4000:4000" ]
    volumes:
      - "./pelias.json:/code/pelias.json"
```

```
  placeholder:
    image: pelias/placeholder:master
    container_name: pelias_placeholder
    user: "${DOCKER_USER}"
    restart: always
    environment: [ "PORT=4100" ]
    ports: [ "4100:4100" ]
    volumes:
      - "./pelias.json:/code/pelias.json"
      - "${DATA_DIR}:/data"
      - "./blacklist/:/data/blacklist"
  whosonfirst:
    image: pelias/whosonfirst:master
    container_name: pelias_whosonfirst
    user: "${DOCKER_USER}"
    volumes:
      - "./pelias.json:/code/pelias.json"
      - "${DATA_DIR}:/data"
      - "./blacklist/:/data/blacklist"
  openstreetmap:
    image: pelias/openstreetmap:relations_bugfix-2019-04-25-
cc778095371c142147e31249947a3b43fb57d46d
    container_name: pelias_openstreetmap
    user: "${DOCKER_USER}"
    volumes:
      - "./pelias.json:/code/pelias.json"
      - "${DATA_DIR}:/data"
      - "./blacklist/:/data/blacklist"
  openaddresses:
    image: pelias/openaddresses:master
    container_name: pelias_openaddresses
    user: "${DOCKER_USER}"
    volumes:
      - "./pelias.json:/code/pelias.json"
      - "${DATA_DIR}:/data"
      - "./blacklist/:/data/blacklist"
  geonames:
    image: pelias/geonames:master
    container_name: pelias_geonames
    user: "${DOCKER_USER}"
    volumes:
      - "./pelias.json:/code/pelias.json"
      - "${DATA_DIR}:/data"
      - "./blacklist/:/data/blacklist"
  csv-importer:
    image: pelias/csv-importer:master
    container_name: pelias_csv_importer
    user: "${DOCKER_USER}"
    volumes:
      - "./pelias.json:/code/pelias.json"
      - "${DATA_DIR}:/data"
      - "./blacklist/:/data/blacklist"
  transit:
    image: pelias/transit:master
    container_name: pelias_transit
    user: "${DOCKER_USER}"
    volumes:
      - "./pelias.json:/code/pelias.json"
      - "${DATA_DIR}:/data"
  polylines:
```

```
    image: pelias/polylines:master
    container_name: pelias_polylines
    user: "${DOCKER_USER}"
    volumes:
      - "./pelias.json:/code/pelias.json"
      - "${DATA_DIR}:/data"
  interpolation:
    image: pelias/interpolation:master
    container_name: pelias_interpolation
    user: "${DOCKER_USER}"
    restart: always
    environment: [ "PORT=4300" ]
    ports: [ "4300:4300" ]
    volumes:
      - "./pelias.json:/code/pelias.json"
      - "${DATA_DIR}:/data"
  pip:
    image: pelias/pip-service:master
    container_name: pelias_pip-service
    user: "${DOCKER_USER}"
    restart: always
    environment: [ "PORT=4200" ]
    ports: [ "4200:4200" ]
    volumes:
      - "./pelias.json:/code/pelias.json"
      - "${DATA_DIR}:/data"
  elasticsearch:
    image: pelias/elasticsearch
    container_name: pelias_elasticsearch
    restart: always
    environment: [ "ES_JAVA_OPTS=-Xmx12g" ]
    ports: [ "9200:9200", "9300:9300" ]
    volumes:
      - "./elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml:ro"
      - "${DATA_DIR}/elasticsearch:/usr/share/elasticsearch/data"
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 65536
        hard: 65536
    cap_add: [ "IPC_LOCK" ]
  fuzzy-tester:
    image: pelias/fuzzy-tester:master
    container_name: pelias_fuzzy_tester
    user: "${DOCKER_USER}"
    restart: "no"
    command: "--help"
    volumes:
      - "./pelias.json:/code/pelias.json"
      - "./test_cases:/code/pelias/fuzzy-tester/test_cases"
```

DOCKER_USER and DATA_DIR in this file are mapped to the corresponding entries in .env. pelias.json is made accessible inside containers in /code/pelias.json. Ports are mapped in the following way: hostport:conainterport. “image:” tells docker from where it has to pull the container image. In this case all images are pulled from the Pelias repository on Docker-Hub. The colon specifies a tag (e.g. master). If no tag is provided, the latest version will be pulled.

With this configuration it is possible to build Europe with the following commands and order (change to Europe project folder first):

```
$ pelias compose pull     → pulls all images defined in docker-compose.yml
$ pelias elastic start    → start Elasticsearch server/container
$ pelias elastic wait     → wait for Elasticsearch to start
$ pelias elastic create   → create Elasticsearch index with pelias mapping
$ pelias download all     → (re) download all data defined pelias.json.
                          Individual (re) downloads are possible too: $ pelias
                          download wof/oa/osm/csv
$ pelias prepare all      → builds all services defined in pelias.json which have
                          a prepare steps. These are polylines (export road
                          network from openstreetmap into polylines format),
                          interpolation (build interpolation sqlite database) and
                          placeholder (build placeholder sqlite database)
$ pelias import all       → (re)import all data (whosonfirst, openaddresses,
                          openstreetmap, polylines, geonames, csv). Individual
                          (re) imports are possible as well:
                          $ pelias import wof/oa/osm/polylines/geonames/csv
$ pelias compose up → start all services defined in docker-compose.yml
```

Afterwards pelias can be queried (e.g. search for postcode 1120 in Belgium):
http://141.59.29.110:4000/v1/search/structured?postalcode=1120&country=be

Result:


```
"type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "geometry": {
                "type": "Point",
                "coordinates": [
                    4.390871,
                    50.899273
                ]
            },
            "properties": {
                "id": "1126113859",
                "gid": "whosonfirst:postalcode:1126113859",
                "layer": "postalcode",
                "source": "whosonfirst",
                "source_id": "1126113859",
                "name": "1120",
                "postalcode": "1120",
                "postalcode_gid": "whosonfirst:postalcode:1126113859",
                "confidence": 1,
                "match_type": "exact",
                "distance": 269.399,
                "accuracy": "centroid",
                "country": "Belgium",
                "country_gid": "whosonfirst:country:85632997",
                "country_a": "BEL",
                "region": "Brussels Capital Region",
                "region_gid": "whosonfirst:region:85681713",
                "region_a": "BRU",
                "continent": "Europe",
                "continent_gid": "whosonfirst:continent:102191581",
                "label": "1120, Belgium"
            },
            "bbox": [
```

```
                4.36532950981,
                50.8813189769,
                4.41362056518,
                50.9139013171
            ]
        }
    ],
```


Giving an exact match (based on whosonfirst data) and a confidence of 1. A complete overview on how to interpret result data can be found here [2].

Documentation on forward and reverse geocoding with Pelias can be found here [3].

2-digit postcodes were imported as a custom layer (“geonamesandpostcodeinfo”, defined in pelias.json) in Pelias. The CSV file has to be copied to the following path (according to docker-compose.yml and .env):
/data/pelias-docker-compose/geonamesandpostcodeinfo/
geonamesandpostcodeinfo2Dpostalcodes.csv
Afterwards the command “pelias import csv” has to be run.

Complete documentation of the CSV importer can be found here [4].

Example:
http://141.59.29.110:4000/v1/search?text=DE81&sources=geonamesandpostcodeinfo

Result:


```
"type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "geometry": {
                "type": "Point",
                "coordinates": [
                    11.6046,
                    48.1331
                ]
            },
            "properties": {
                "id": "1141",
                "gid": "geonamesandpostcodeinfo:postalcode:1141",
                "layer": "postalcode",
                "source": "geonamesandpostcodeinfo",
                "source_id": "1141",
                "name": "DE81",
                "confidence": 1,
                "match_type": "exact",
                "distance": 690.624,
                "accuracy": "centroid",
                "country": "Germany",
                "country_gid": "whosonfirst:country:85633111",
                "country_a": "DEU",
                "region": "Bayern",
                "region_gid": "whosonfirst:region:85682571",
                "region_a": "BY",
                "macrocounty": "Oberbayern",
                "macrocounty_gid": "whosonfirst:macrocounty:404227567",
                "county": "München",
                "county_gid": "whosonfirst:county:102063261",
```

```
                "county_a": "MN",
                "locality": "München",
                "locality_gid": "whosonfirst:locality:101748479",
                "neighbourhood": "Haidhausen",
                "neighbourhood_gid": "whosonfirst:neighbourhood:85905613",
                "continent": "Europe",
                "continent_gid": "whosonfirst:continent:102191581",
                "label": "DE81, München, Germany"
            }
        }
    ],
    "bbox": [
        11.6046,
        48.1331,
        11.6046,
        48.1331
    ]
}
```


Installation from scratch

Dependecies:

1. Node.js:

Version 8 or newer required, version 10 recommended

Ubuntu or Debian:

Node.js v11.x:

```
# Using Ubuntu

curl -sL https://deb.nodesource.com/setup_11.x | sudo -E bash -
sudo apt-get install -y nodejs

# Using Debian, as root

curl -sL https://deb.nodesource.com/setup_11.x | bash -
apt-get install -y nodejs
```

Node.js v10.x:

```
# Using Ubuntu

curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -
sudo apt-get install -y nodejs

# Using Debian, as root

curl -sL https://deb.nodesource.com/setup_10.x | bash -
apt-get install -y nodejs
```

CentOS:

NodeJS 11.x

```
curl -sL https://rpm.nodesource.com/setup_11.x | bash -
```

NodeJS 10.x

```
curl -sL https://rpm.nodesource.com/setup_10.x | bash -
```

Optional: install build tools

To compile and install native addons from npm you may also need to install build tools:

```
yum install gcc-c++ make
# or: yum groupinstall 'Development Tools'
```

2. Elasticsearch:

Version 2.4 or 5.6

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" | sudo tee -a
/etc/apt/sources.list.d/elastic-5.x.list
sudo apt update && sudo apt upgrade
sudo apt install apt-transport-https uuid-runtime pwgen openjdk-8-jre-headless
sudo apt-get update
sudo apt update
sudo apt install elasticsearch
mkdir /elasticsearch
mkdir /elasticsearch/data
mkdir /elasticsearch/logs
```

make sure elasticsearch has rights to access the folders:

```
sudo chown -R elasticsearch:elasticsearch /elasticsearch
```

if you get Errors like:

```
ERROR Null object returned for RollingFile in Appenders
```

that most likely means that elasticsearch doesn't have permissions to access the logs and data folders.

After the installation, a default configuration file will be populated to

/etc/elasticsearch/elasticsearch.yml

Most lines are commented out, edit the file to tweak and tune the configuration.

E.g, you can set correct cluster name for your applications:

```
cluster.name: my-application
```

Recommended Settings:

```
cluster.name: pelias-el-search
path.data: /elasticsearch/data
path.logs: /elasticsearch/logs
network.host: 127.0.0.1
http.port: 9200
```

Note that the default minimum memory set for JVM is 2gb, if your server has small memory size, change this value:

```
$ sudo vim /etc/elasticsearch/jvm.options
```

Change:

```
-Xms2g
-Xmx2g
```

And set your values for minimum and maximum memory allocation. E.g to set values to 512mb of ram, use:

```
-Xms512m
-Xmx512m
```

After you have modified the configuration, you can start Elasticsearch:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable elasticsearch.service
$ sudo systemctl restart elasticsearch.service
```

Check status:

```
$ sudo systemctl status elasticsearch.service
```

```
● elasticsearch.service - Elasticsearch
Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; disabled;
vendor preset: enabled)
Active: active (running) since Sun 2019-05-01 10:39:54 UTC; 18s ago
Docs: http://www.elastic.co
Process: 14314 ExecStartPre=/usr/share/elasticsearch/bin/elasticsearch-
systemd-pre-exec (code=exited, status=0/SUCCESS)
Main PID: 14325 (java)
Tasks: 38 (limit: 2362)
CGroup: /system.slice/elasticsearch.service
└─14325 /usr/bin/java -Xms512m -Xmx512m -XX:+UseConcMarkSweepGC -
XX:CMSInitiatingOccupancyFraction=75 -XX:+UseCMSInitiatingOccupancyOnly -X
```

That's all for the installation of Elasticsearch 5.x on Ubuntu 18.04 LTS (Bionic Beaver) Linux.

3. SQLite:

Version 3.11 or newer

```
sudo apt-get update
sudo apt-get install sqlite3
sqlite3 --version
sudo apt-get install sqlitebrowser
```

4.Libpostal:

```
sudo apt-get install curl autoconf automake libtool pkg-config
cd /
git clone https://github.com/openvenues/libpostal
cd libpostal
./bootstrap.sh
./configure --datadir=[...some dir with a few GB of space...]
make -j4
sudo make install
```

```
# On Linux it's probably a good idea to run
sudo ldconfig
```

Pelias:

Data:

Download all the rawdata you need and copy it into respective folders such as

```
"/data/rawdata/whosonfirst",
"/data/rawdata/geonames",
"/data/rawdata/openadresses",
"/data/rawdata/openstreetmap",
"/data/rawdata/polylines"
```

Pelias:

```
mkdir /pelias
cd /pelias
for repository in schema whosonfirst geonames openaddresses openstreetmap
polylines api placeholder interpolation pip-service; do
      git clone https://github.com/pelias/${repository}.git # clone from Github
      pushd $repository > /dev/null                         # switch into importer
directory
      npm install                                           # install npm
dependencies
      popd > /dev/null                                      # return to code
directory
done
```

Create a file "config.json" in the home directory (~/) of the pelias user. Paste following into pelias.json:

```
{
  "esclient": {
    "apiVersion": "5.6",
    "keepAlive": true,
    "requestTimeout": "120000",
    "hosts": [{
      "env": "development",
      "protocol": "http",
      "host": "localhost",
      "port": 9200
    }],
    "log": [{
      "type": "stdio",
      "json": false,
      "level": [ "error", "warning" ]
    }]
  },
  "elasticsearch": {
    "settings": {
      "index": {
        "number_of_replicas": "0",
        "number_of_shards": "5",
        "refresh_interval": "1m"
```

```
      }
    }
  },
  "interpolation": {
    "client": {
      "adapter": "null"
    }
  },
  "dbclient": {
    "statFrequency": 10000
  },
  "api": {
    "accessLog": "common",
    "textAnalyzer": "libpostal",
    "host": "http://pelias.mapzen.com/",
    "indexName": "pelias",
    "version": "1.0",
    "targets": {
      "auto_discover": false,
      "canonical_sources": ["whosonfirst", "openstreetmap", "openaddresses",
"geonames"],
      "layers_by_source": {
        "openstreetmap": [ "address", "venue", "street" ],
        "openaddresses": [ "address" ],
        "geonames": [
          "country", "macroregion", "region", "county", "localadmin", "locality",
"borough",
          "neighbourhood", "venue"
        ],
        "whosonfirst": [
          "continent", "empire", "country", "dependency", "macroregion", "region",
"locality",
          "localadmin", "macrocounty", "county", "macrohood", "borough",
"neighbourhood",
          "microhood", "disputed", "venue", "postalcode", "continent", "ocean",
"marinearea"
        ]
      },
      "source_aliases": {
        "osm": [ "openstreetmap" ],
        "oa":  [ "openaddresses" ],
        "gn":  [ "geonames" ],
        "wof": [ "whosonfirst" ]
      },
      "layer_aliases": {
        "coarse": [
          "continent", "empire", "country", "dependency", "macroregion", "region",
"locality",
          "localadmin", "macrocounty", "county", "macrohood", "borough",
"neighbourhood",
          "microhood", "disputed", "postalcode", "continent", "ocean",
"marinearea"
```

```
      ]
    }
  }
},
"schema": {
  "indexName": "pelias"
},
"logger": {
  "level": "debug",
  "timestamp": true,
  "colorize": true
},
"acceptance-tests": {
  "endpoints": {
    "local": "http://localhost:3100/v1/",
    "dev-cached": "http://pelias.dev.mapzen.com.global.prod.fastly.net/v1/",
    "dev": "http://pelias.dev.mapzen.com/v1/",
    "prod": "http://search.mapzen.com/v1/",
    "prod-uncached": "http://pelias.mapzen.com/v1/",
    "prodbuild": "http://pelias.prodbuild.mapzen.com/v1/"
  }
},
"imports": {
  "adminLookup": {
    "enabled": true,
    "maxConcurrentRequests": 100,
    "usePostalCities": false
  },
  "blacklist": {
    "files": []
  },
  "csv": {
  },
  "geonames": {
    "datapath": "/data/rawdata/geonames",
    "countryCode": "ALL"
  },
  "openstreetmap": {
    "datapath": "/data/rawdata/openstreetmaps",
    "leveldbpath": "/tmp",
    "import": [{
      "filename": "europe-latest.osm.pbf"
    }]
  },
  "openaddresses": {
    "datapath": "/data/rawdata/openaddresses",
    "files": []
  },
  "polyline": {
    "datapath": "/data/rawdata/polylines",
    "files": []
  },
```

```
    "whosonfirst": {
      "datapath": "/data/rawdata/whosonfirst",
      "importVenues": false
    }
  }
}
```

Add the pelias.json to the PATH variable by adding the followinbg line at the bottom of the .bashrc-file:

```
export PATH=$PATH:PELIAS_CONFIG=/home/<username>/pelias.config
```

Set up Elasticsearch Schema:

```
cd /pelias/schema  # assuming you have just run the bash snippet to
        download the repos from earlier
./bin/create_index
```

Run the Importers:

For each importer (openadresses, openstreetmaps, whosonfirst, geonames and polylines) navigate into their folders under

```
cd /pelias/<importer_folder>
```

and execute the importer via

```
npm start
```

In case you want to delete the imported data and restart from import phase, run the following:

```
# !! WARNING: this will remove all your data from pelias!!
node scripts/drop_index.js    #it will ask for confirmation first
./bin/create_index
```

Install and start the pelias services:

Add the following to the bottom of the pelias.json file in your home directory. This will tell the pelias API to use all the services running locally and on their default ports.

```
{
  "api": {
    "services": {
      "placeholder": {
        "url": "http://localhost:3000"
      },
      "libpostal": {
        "url": "http://localhost:8080"
      },
      "pip": {
        "url": "http://localhost:3102"
      },
      "interpolation": {
        "url": "http://localhost:3000"
      }
    }
  }
```

```
}
```

Start the pelias API:

To start the pelias API navigate into folder

```
cd /pelias/api
```

and execute

```
npm start
```

Geocoding with pelias:

Pelias should now be up and running and will respond to your queries.

For a quick check, a request to http://localhost:3100 should display a link to the documentation for handy reference.

Here are some queries to try:

http://localhost:3100/v1/search?text=london: a search for the city of London.

http://localhost:3100/v1/autocomplete?text=londo: another query for London, but using the autocomplete endpoint which supports partial matches and is intended to be sent queries as a user types (note the query is for londo but London is returned)

http://localhost:3100/v1/reverse?point.lon=-73.986027&point.lat=40.748517: a reverse geocode for results near the Empire State Building in New York City.

2-digit postcodes

CSV Data provided by geonames.org and postcode.info was used as basis for calculating 2-digit postcodes for European countries.

- https://download.geonames.org/export/zip/allCountries.zip [5]
- postcode.info was scraped and provided as a CSV file by a fellow student

In the first iteration 2-digit postcodes were calculated from geonames data [6]:

- allCountries.txt is the CSV downloaded from geonames
- calculate2DpostCodeFromGeonames.py calculates 2-digit postcodes and creates a new CSV file
- 1file.py in case codes for only one country have to be calculated
- geonames2Dpostalcodes.csv contains the 2-dgit postcodes in a file which can be imported into Pelias (output of calculate2DpostCodeFromGeonames.py):

```
DE80,geonames2d,80,postalcode,48.1615,11.5509
DE81,geonames2d,81,postalcode,48.1254,11.5726
DE82,geonames2d,82,postalcode,47.911,11.2502
DE83,geonames2d,83,postalcode,47.8713,12.2803
DE84,geonames2d,84,postalcode,48.4086,12.4327
DE85,geonames2d,85,postalcode,48.4174,11.6308
```

In the second iteration both sources were combined after having done some preparation steps on the postcode.info CSV file [7]:

- calculate2DpostcodeFromGeonamesAndPostcodeinfo.py is the script which calculates 2-digit postcodes. It expects a single CSV file (merge of geonames and postcode.info).
- geonamesandpostcodeinfo2Dpostalcodes.csv is the generated file for Pelias:

```
name,source,country,postalcode,layer,lat,lon
DE80,geonamesandpostcodeinfo,DE,80,postalcode,48.1512,11.5938
DE81,geonamesandpostcodeinfo,DE,81,postalcode,48.1331,11.6046
DE82,geonamesandpostcodeinfo,DE,82,postalcode,47.9419,11.2759
DE83,geonamesandpostcodeinfo,DE,83,postalcode,47.888,12.2627
DE84,geonamesandpostcodeinfo,DE,84,postalcode,48.4367,12.4206
DE85,geonamesandpostcodeinfo,DE,85,postalcode,48.4171,11.6294
```

Routing engines

The Pelias API and its associated services are only suited for the purpose of geocoding, meaning to retrieve coordinates (latitude and longitude) for a given address or postal code, or performing reverse geocoding (finding the nearest known address or postal code for the given coordinates.

If we want to find the shortest or fastest route between two given addresses or coordinates, we would have to use Pelias in connection with a routing engine. We would feed our two addresses into Pelias and Pelias would give us the coordinates to these two addresses. Afterwards we feed these coordinates into the routing engine and would receive the best/shortest route between the two points according to some metrics inside the routing engine.

The senior team used the routing engine Graphhopper in connection with their geocoding service Nominatim. Graphhopper as you will see later in this report is a very good and fast routing engine, however the developers of the Pelias service recommend using the routing engine Valhalla which is developed by the same company (Mapzen) as Pelias and therefore has better service interoperability with Pelias than any other routing engine.

Comparison of routing engines [8]

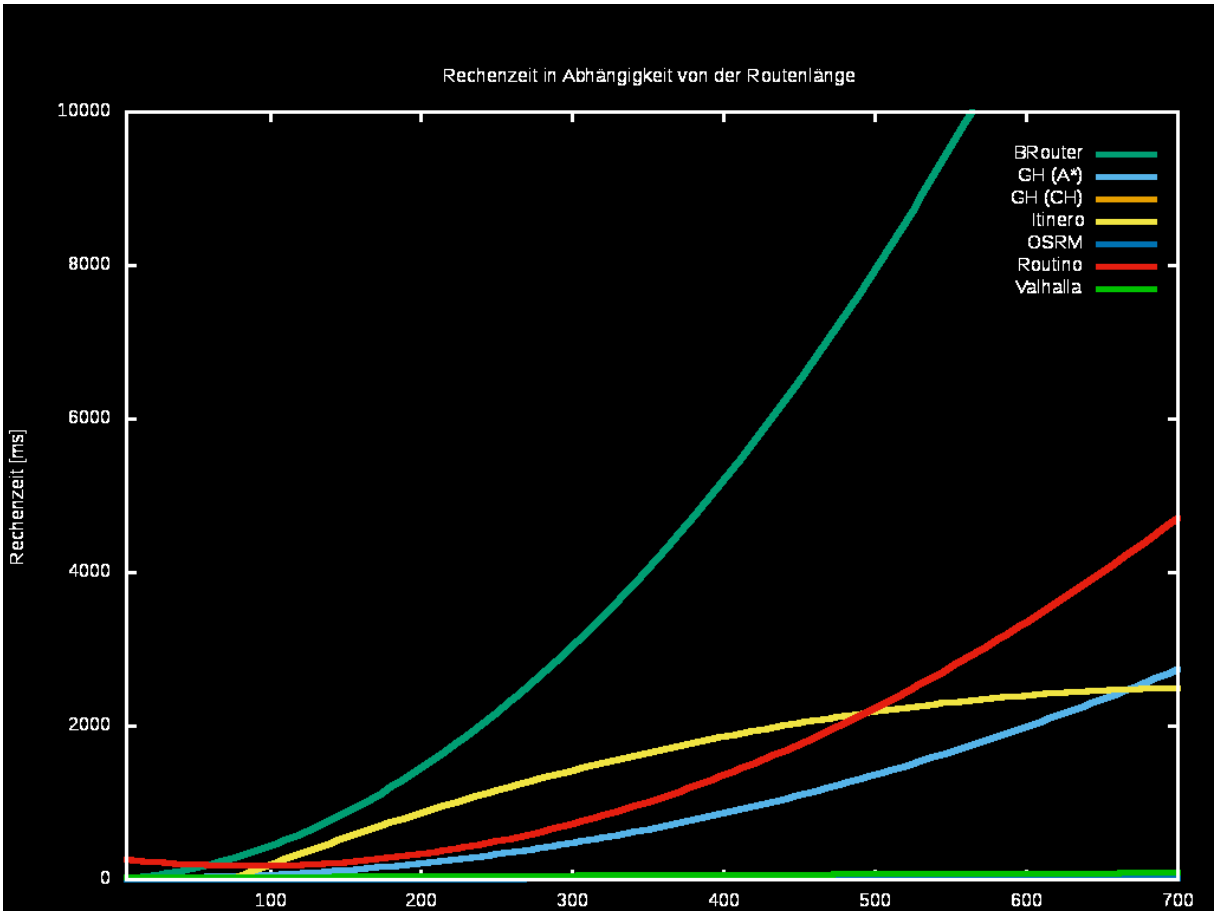

Figure 2: Comparison of Routing Engines 1 from [8]

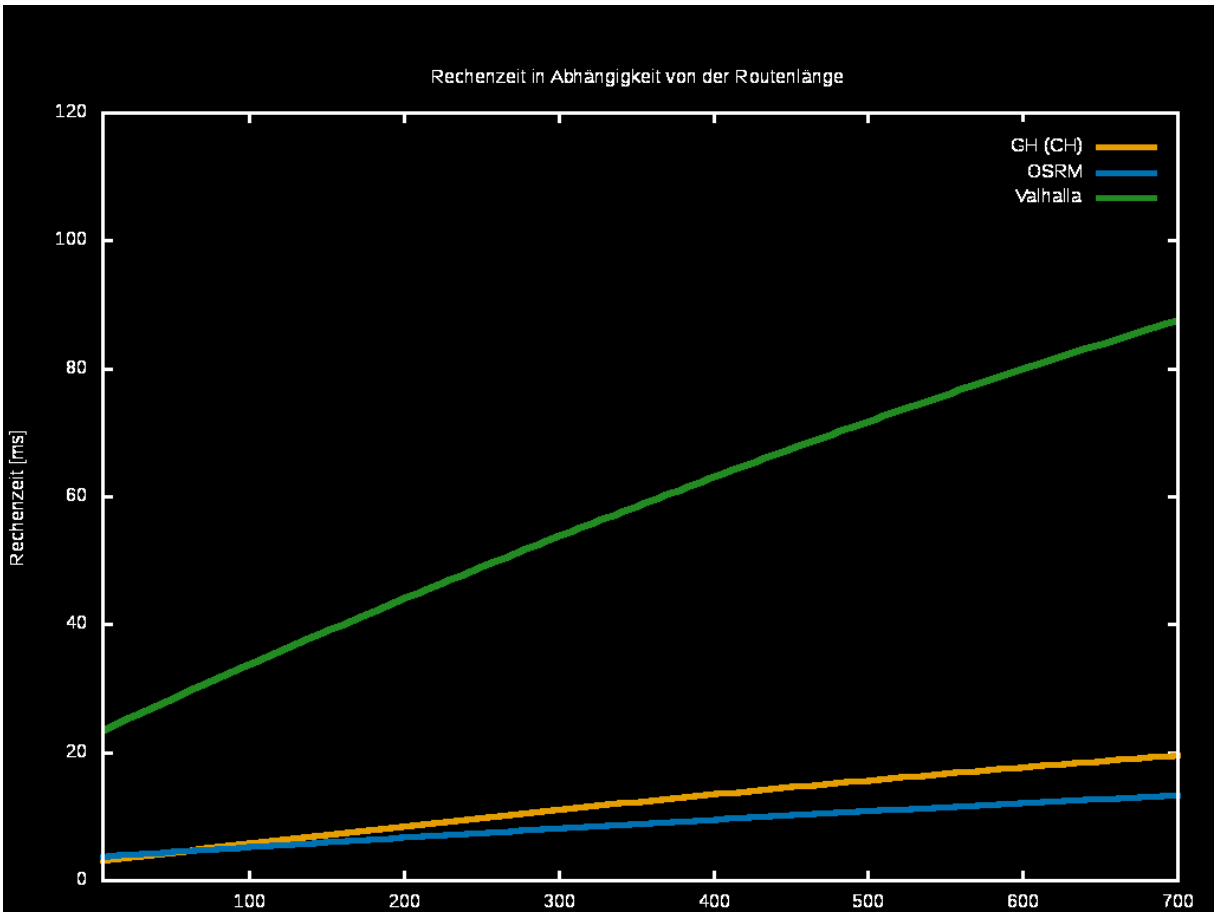

Figure 3: Comparison of Routing Engines 2 from [8]

Because Graphhopper, OSRM (Open Source Routing Machine) and Valhalla seem to have the best performance we took a closer look at them.

Comparison criteria

- License
- Language, Operating System
- Algorithm used
- Documentation and setup
- Routingfeatures
- Performance
- Special features

Graphhopper

License:	Apache-License (proprietary in parts)
OS:	Java (also Android, iOS)
Algorithm:	Contraction Hierarchies, Dijkstra/A*, Hybrid
Documentation & setup:	Good documentation, ‘quick start’
Routingfeatures:	Turn restriction (A*), guidepost, alternatives, height data optional
Special features:	Track Matching, cost != time, TSP with jsprit

Valhalla

License:	MIT license
OS:	C++, Apple/Linux
Algorithm:	A* with individual improvements

Documentation & setup:	Good documentation, ‘quick start’, ubuntu repository with web-frontend
Routingfeatures:	Turn restriction, guidepost, height data optional
Special features:	Tile-based data storage, dynamic cost, matrix, isochrones, intermodal, Designed for working with OpenStreetMap

OSRM (Open Street Routing Machine)

License:	BSD license
OS:	C++ (NodeJS), Apple/Linux/WIndows
Algorithm:	Contraction Hierarchies
Documentation & setup:	Good documentation, setup with docker or self-compiled
Routingfeatures:	Turn restriction (A*), driving lanes, guidepost, alternatives, no height data
Special features:	Matrix, track matching, TSP, data tiles, cost != time

However OSRM needs a lot of RAM and disc memory. See this extract from [9]:

Pre-Processing
For the car profile you will need

- around 175 GB of RAM for pre-processing
- around 280 GB of STXXL disk space

you'll also need 35 GB for the planet .osm.pbf file, and 40-50 GB for the generated datafiles.
For the foot profile the latest number we have are about 248 GB of RAM. Everything else is proportionally larger.
Runtime
For the car profile you will need

- around 64GB of RAM

We basically just load all the files into memory, so whatever the output file size from pre-processing - that's roughly how much RAM you'll need.

Conclusion

OSRM is the fastest routing engine on the open-source market. But because of the very high memory requirements of OSRM it is not suitable for the use-case of our project and the cost/benefit-factor is too low. Valhalla would be a good alternative to Graphhopper, because it is compared to other routing engines as fast as Graphhopper and is designed to work with OpenStreetMap-data and also recommended by the Pelias developers to be used in connection with Pelias as a geocoder. However, in this early state of the project Graphhopper totally fits all the needs and there wouldn’t be much sense in replacing Graphhopper with Valhalla.

Valhalla

Valhalla was installed and configured according to the official documentation on Github [10].

Tiles and polylines were calculated using the same openstreetmaps pbf file (Europe) which was already used for Pelias. Routing can be achieved by querying Valhalla's api:

```
curl http://141.59.29.110:8002/route --data
'{"locations":[{"lat":48.1331,"lon":11.6046,"type":"break"},{"lat":47.9419,"lon":1
1.2759,"type":"break"}],"costing":"auto","directions_options":{"units":"km"}}' |
jq '.'
```

Result:


```
    "summary": {
      "max_lon": 11.605507,
      "max_lat": 48.133167,
      "time": 2397,
      "length": 38.536,
      "min_lat": 47.943157,
      "min_lon": 11.260186
    },
    "locations": [
      {
        "original_index": 0,
        "type": "break",
        "lon": 11.6046,
        "lat": 48.133099,
        "side_of_street": "right"
      },
      {
        "original_index": 1,
        "type": "break",
        "lon": 11.2759,
        "lat": 47.941898,
        "side_of_street": "right"
      }
```


The complete output and routing instructions are in Appendix 1.

List of Figures

Bibliography

[1] J. o. Simioni, 24 04 2018. [Online]. Available: https://raw.githubusercontent.com/pelias/pelias/master/img/Pelias%20Architecture.png. [Accessed 14 06 2019].

[2] J. o. Simioni, “https://github.com/pelias,” 10 08 2018. [Online]. Available: https://github.com/pelias/documentation/blob/master/result_quality.md. [Accessed 21 06 2019].

[3] J. o. Simioni, “https://github.com/pelias/documentation,” 2 11 2018. [Online]. Available: https://github.com/pelias/documentation. [Accessed 21 06 2019].

[4] J. o. Simioni, “https://github.com/pelias/csv-importer,” 17 05 2019. [Online]. Available: https://github.com/pelias/csv-importer. [Accessed 21 06 2019].

[5] Unxos GmbH, Weingartenstrasse 8, 8708 Maennedorf, Switzerland, “https://download.geonames.org/,” Unxos GmbH, Weingartenstrasse 8, 8708 Männedorf, Switzerland, [Online]. Available: https://download.geonames.org/export/zip/allCountries.zip. [Accessed 21 06 2019].

[6] S. Dechant, “https://github.com/dataBikeHsUlm/juniors/tree/master/geonames_2D_postalcodes,” 22 05 2019. [Online]. Available: https://github.com/dataBikeHsUlm/juniors/tree/master/geonames_2D_postalcodes. [Accessed 21 06 2019].

[7] S. Dechant, “https://github.com/dataBikeHsUlm/juniors/tree/master/peliasimport,” 04 06 2019. [Online]. Available: https://github.com/dataBikeHsUlm/juniors/tree/master/peliasimport. [Accessed 21 06 2019].

[8] F. Ramm, "https://www.geofabrik.de/," 23 03 2017. [Online]. Available: https://www.geofabrik.de/media/2017-03-23-Routing-Engines_fuer_OSM.pdf. [Accessed 20 04 2019].

[9] D. J. H., "https://github.com/Project-OSRM/," 05 07 2017. [Online]. Available: https://github.com/Project-OSRM/osrm-backend/wiki/Disk-and-Memory-Requirements. [Accessed 13 06 2019].

[10] G. g. Knisely, "https://github.com/valhalla/valhalla," [Online]. Available: https://github.com/valhalla/valhalla. [Accessed 01 04 2019].

Appendixes

Appendix 1


```
{
  "trip": {
    "language": "en-US",
    "status": 0,
    "units": "kilometers",
    "status_message": "Found route between points",
    "legs": [
      {
        "shape":
"}ayxzAmghcU|@sQFqGpBGdT}@hHCvIL|DLtOVlJCvXIpMEl^@rQ?nXCfIo@lEmA`CsA~CuChCwCtDcGlA
_B~g@b|@\\h@`ClErAvCrArC`C~EjAxBxQb[jQ|ZnH|JtExF|@nAbFhI|EjIdZjh@rPtYjPvXhWxd@pHvM
nDtGlEbIbF~I`CjDjBzCxBrAnC`AxBnEpBjEtEzIhCrFd@x@~Wti@|Txg@~Nf[xBnFl@zAjA|BrAbE~C`S
Fn@`CjP\\jBt@dGdEj[bAfIxBdN`CnIrA|Fl@bCvH|_@fIrc@tEf[hCjUdEba@hCdYxBlXL`BbBtUd@vGh
BxWdAtLzAdUlJzqAL~BjBdTpBlTrAzMhBtMjBjMpBlLtDfRvDbP|@nEd@|D?pDW|C]~B]xBWx@fDjCpBhB
nOdMpB~Al@b@xB~Az@l@VVxBdBjFjEhCnB|N|JxGjEzAfAfIvFFDjAz@vChDhH|IdKvKbFtEtEtDjFdEfJ
lHj@f@t@z@NTrAvB\\^NTd@r@T\\NXl@z@d@dALTVV~CnDl@r@r@t@xHpHvHrG~DbC~WrWpBbAtEbDrGpG
|IxJrKhLdAjAbAhAbAlAlObQhCtClOvQdApAz@dCjBtD~HxJbAnAhBvBbBbBbBpB|IfKzFtGvCtDbB`Be@
pA]pAe@nB_@rCs@zIWnGaC~h@e@vJUbIGhW?nTFpHrApJLdCN`CThBd@zBl@tBdA~Bz@jBrAnBbBlAtDrA
pHOfDKbKlD`H`CtErA~Cx@hCt@hCPfCBhC@bFBdA@r@?bBDdUNfXe@fDA~SdE~MdDvCx@~C`ArBd@\\Fl@
ZfCv@dFlAze@|KpB`@tEt@u@zGU`CcGlg@kAhKyBvSe@pF_@pFUxFGfEGdEFpFV~Fd@nFz@bFdAtExAxEb
B|D~CfFhBbCrBvB~CzBvCtBzFzC|DbCxChC~BpChCjDvCpFrAvCbBxExBrIz@nFt@`I\\jKEvIe@dKmAnK
qBvJ{AtFyBzGwCxGiBpDyBzDsBvCmDvEsA~AoDbDwCpCiCdC}@p@uI~GcAr@aCbBiH~FkGnEmDtDyB|DkB
nEgC~GqCnKcGzYkF~]oCdUaCzWkAzN}@bMkAnTe@fJm@|KGzB?bJUnLWpKMrK?|ODlL?rM|@dHN~QTtSl@
rSt@hSxAnZjBhZbF|t@fNrrB`Ct^pBx\\rAp\\t@fU\\hUNbO?r\\Gnc@?bOUvjA{AlvEpMjdDxLv~Cz@l
[l@nOt@vPpBtZzA`RpBhRzA`MhB`LhCnM`H`YxLrc@pHn[pBpKpBdMEbBpB`OrAfMz@lLdA|L`Bbe@VtRL
`R^``Bz@pzAd@~z@d@r]bAl_@pClo@`Cti@bGjaArEb{@xH|uAr@jQf@h^g@pbCUnkCiBfk@m@rk@WzaB?
hK]nq@GvDM~GOnBm@vF_DlF{KpNkLfNaB|AmFlNqAhJGp@?rCl@dGrAtGnR|f@xGdOzGlOjUrj@vSbg@bL
hYvHdPpHbKfvAvkDpdB~fE|gA~nCh`B~~DlJpUzUbl@tYpr@nm@f{Ap]|{@pcBthEpbD|bIp]|v@rP|XbP
jTjVfUnXfQbVnKnYxFpa@nC|hANzcAq@pk@a@fc@B`aF}B|m@c@tkGaCfm@^nIBzUnAvShA|TpC~MvAxa@
vGp\\rInc@bNhb@tOj[~M|XhPn^tUxq@fi@faFbiExLnKphGzhFvwAvpA|jCx_Cfh@xg@fNhNpf@ho@dKn
Nvl@~~@zPp\\fIpNrQd_@vW|n@hN~^tOtb@xVfv@pv@b_CjiAn|CnI|T~w@vlBp`ApwBlJtR~|@tlB`MfW
flA~vBv{@vyAn^zm@blCl}DfqBr~Cpe@f`Ar_AllB~h@niA|Tbj@lrAvkDjiAhdDzy@blBd~@v~Abe@bo@
r[t^zVxWv\\r\\dz@tq@bz@jg@xjAdk@biAji@h\\`QlKtFpL~GxjAzn@rdAxp@jtAx_A~fA~z@ld@f_@r
ZhVre@zb@pf@pd@~lAjiAdZrWh\\bXfSzNlT~O~^|UrZbQ~M|HbG`IxHvEp\\bR`f@hTbe@pSze@bSjk@p
V`M|FtO~IzPrKjQbNvRdQvNvO`R~T`MrQfOdV~MfWjLfWzK~XbK`Z`Xbz@f]tfAtO~e@jQdd@rAxDpa@zl
ArQre@hRje@~MpZdOv\\`Xvi@ro@nqAb~A|xC|lB~{Drt@xqBxMnc@vR~p@zUxeAbRjnArF|d@dEf]|Jbx
A~Bbz@rBts@`BdjCjAprA`DtjApBfm@vHvqA`Cz]bFtm@lEhd@hRpiBrQ|}ArPteBfNhzA`Cbb@bGdz@bG
x}@|NddDpM`mCnC|p@hC|t@bApXlAl`@bA|[z@fXVfJ`Bnn@`Cx~@bBtu@z@z^~Cn_BbBvgAd@p\\NbM`B
jsAdApYfCdt@VtOxBjbALpKe@nQkA|LqBzLoDbMaBhFsBpEgCxF_DdGwb@vt@i]pk@iCfFiG|LuJnVcFpX
cBrYcAzYm@r^_@bScAld@FzI\\vJd@fEjAnFrAfFrAbDrA`D~N|ZbKvSdA`Bv]fl@lExG~b@`RrKrIjFtE
tEhEjAt@dFhC~g@nRtEhBbQxLlOzX~CpFlEfIhCjEd@p@dTlXjFlG`HnGp\\lY~IvIrKrTnNd]rQzq@z@z
CTdAdFlRnDzJfHnMz[be@hBrCzAfCxp@nn@to@xc@bj@ff@lZ|YjP`VfObXdEzI~HzSdFnW`BtItE`^hC~
RbAjVfIpq@z@lHbGfh@fD|LbB~MnCbSjAjKlJxv@xBtSdFzc@hBjQ|End@zFbb@vNzv@nRnr@zVvf@fm@t
z@|D~Ep\\ja@lEhFnh@dr@p\\rd@bp@d{@dZpd@`B`CbR~VzT|e@fYxg@`S`\\bPtVbVz\\dPbSfXlXd_@
b_@zKdOdErGfDfHm@zEFfFl@vEl@`CxBzDhBvAxCbA~CAvCeAhCcCzEbC~IjHbFdErFfI~DtF|IhMt_@xp
@`eB`oDh\\ft@`\\`u@`S`f@nHrRhR`h@`HnTxHhU~BjHzA|E|Y~dAr[xpA|Ot{@fDzQnDpTxB`NlIdh@j
Gt_@xVjzApClPvH~c@jBhLzEvU~DdS~C`Q|N`s@`XphAtY~`ArK~]ze@nzAjFrP`IlTj[z{@j`@jfAhBfF
lYbp@tUvc@nIjNhRfYbUjWtUxVnSfOxVbP~NtIhWxK~]lMrU|E|YlDvS^nTb@dd@k@|bAuFbk@nC`WpDr[
fKzJdF`MrGxRzLvb@~a@vIlKtE~FxLbRjFrHhHhMnIhN`]dw@lTlp@`Mff@hMzm@nDhThRtsA`M~j@vMfb
@`R|i@zQ~a@tN|WvSx\\j`@vg@rQbXdPdYtIrSrGxTrK~]x\\pbAdYtj@fTlXnStR|^lTld@bQtc@rQjLn
GbLdGvNvMfr@nx@~HpJl^ld@~h@rp@`Wb\\fYpWnMpJdUdNzAl@|DdB~h@~SjFfSbApFxCnFvCrC|NlJ|s
@tVvNpGpG|ExB_AxCs@pGXfNTtEo@`HmCdEiErAqCbB{PLmPNqKd@cc@zAyOnCqNbQso@zKcc@t@eFTcFE
eEWqEaCkOyVcnA_OacAoMwgAsB_f@t@mn@e@sLiBeS}@kNOaMVeNfC{OfOk_@lEkQ~CkTrKyj@tOin@rKs
n@lAcMd@aFr@{XGqSe@oMEyA",
        "summary": {
          "max_lon": 11.605507,
```

```
          "max_lat": 48.133167,
          "time": 2397,
          "length": 38.536,
          "min_lat": 47.943157,
          "min_lon": 11.260186
        },
        "maneuvers": [
          {
            "travel_type": "car",
            "street_names": [
              "Kirchenstraße"
            ],
            "verbal_pre_transition_instruction": "Drive east on Kirchenstraße for
30 meters. Then Turn right onto Elsässer Straße.",
            "instruction": "Drive east on Kirchenstraße.",
            "end_shape_index": 2,
            "type": 2,
            "time": 15,
            "verbal_multi_cue": true,
            "length": 0.033,
            "begin_shape_index": 0,
            "travel_mode": "drive"
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Turn right onto Elsässer
Straße.",
            "verbal_transition_alert_instruction": "Turn right onto Elsässer
Straße.",
            "length": 0.424,
            "instruction": "Turn right onto Elsässer Straße.",
            "end_shape_index": 21,
            "type": 10,
            "time": 107,
            "verbal_post_transition_instruction": "Continue for 400 meters.",
            "street_names": [
              "Elsässer Straße"
            ],
            "begin_shape_index": 2
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Turn right onto Orleansstraße.",
            "verbal_transition_alert_instruction": "Turn right onto
Orleansstraße.",
            "length": 0.887,
            "instruction": "Turn right onto Orleansstraße.",
            "end_shape_index": 57,
            "type": 10,
            "time": 149,
            "verbal_post_transition_instruction": "Continue for 900 meters.",
            "street_names": [
              "Orleansstraße"
            ],
            "begin_shape_index": 21
          },
          {
            "travel_type": "car",
```

```
            "verbal_pre_transition_instruction": "Continue on Auerfeldstraße for
200 meters.",
            "verbal_transition_alert_instruction": "Continue on Auerfeldstraße.",
            "length": 0.161,
            "instruction": "Continue on Auerfeldstraße.",
            "end_shape_index": 68,
            "type": 8,
            "time": 42,
            "street_names": [
              "Auerfeldstraße"
            ],
            "begin_shape_index": 57,
            "travel_mode": "drive"
          },
          {
            "travel_type": "car",
            "verbal_pre_transition_instruction": "Continue on Welfenstraße for 700
meters.",
            "verbal_transition_alert_instruction": "Continue on Welfenstraße.",
            "length": 0.708,
            "instruction": "Continue on Welfenstraße.",
            "end_shape_index": 100,
            "type": 8,
            "time": 105,
            "street_names": [
              "Welfenstraße"
            ],
            "begin_shape_index": 68,
            "travel_mode": "drive"
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Turn left onto Regerstraße.",
            "verbal_transition_alert_instruction": "Turn left onto Regerstraße.",
            "length": 0.155,
            "instruction": "Turn left onto Regerstraße.",
            "end_shape_index": 113,
            "type": 15,
            "time": 18,
            "verbal_post_transition_instruction": "Continue for 200 meters.",
            "street_names": [
              "Regerstraße"
            ],
            "begin_shape_index": 100
          },
          {
            "travel_type": "car",
            "verbal_pre_transition_instruction": "Continue on Tegernseer
Landstraße for 600 meters.",
            "verbal_transition_alert_instruction": "Continue on Tegernseer
Landstraße.",
            "length": 0.631,
            "instruction": "Continue on Tegernseer Landstraße.",
            "end_shape_index": 167,
            "type": 8,
            "time": 77,
            "street_names": [
              "Tegernseer Landstraße"
            ],
```

```
            "begin_shape_index": 113,
            "travel_mode": "drive"
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Turn right onto Ichostraße.",
            "verbal_transition_alert_instruction": "Turn right onto Ichostraße.",
            "length": 0.183,
            "instruction": "Turn right onto Ichostraße.",
            "end_shape_index": 179,
            "type": 10,
            "time": 27,
            "verbal_post_transition_instruction": "Continue for 200 meters.",
            "street_names": [
              "Ichostraße"
            ],
            "begin_shape_index": 167
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_multi_cue": true,
            "verbal_pre_transition_instruction": "Bear left to stay on Ichostraße.
Then Bear left onto Martin-Luther-Straße.",
            "verbal_transition_alert_instruction": "Bear left to stay on
Ichostraße.",
            "length": 0.045,
            "instruction": "Bear left to stay on Ichostraße.",
            "end_shape_index": 186,
            "type": 16,
            "time": 6,
            "verbal_post_transition_instruction": "Continue for 50 meters.",
            "street_names": [
              "Ichostraße"
            ],
            "begin_shape_index": 179
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Bear left onto Martin-Luther-
Straße.",
            "verbal_transition_alert_instruction": "Bear left onto Martin-Luther-
Straße.",
            "length": 0.347,
            "instruction": "Bear left onto Martin-Luther-Straße.",
            "end_shape_index": 211,
            "type": 16,
            "time": 46,
            "verbal_post_transition_instruction": "Continue for 300 meters.",
            "street_names": [
              "Martin-Luther-Straße"
            ],
            "begin_shape_index": 186
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
```

```
            "verbal_pre_transition_instruction": "Continue on Tegernseer
Landstraße for 100 meters. Then Turn right onto Candidstraße.",
            "verbal_transition_alert_instruction": "Continue on Tegernseer
Landstraße.",
            "length": 0.121,
            "instruction": "Continue on Tegernseer Landstraße.",
            "end_shape_index": 219,
            "type": 8,
            "time": 13,
            "verbal_multi_cue": true,
            "street_names": [
              "Tegernseer Landstraße"
            ],
            "begin_shape_index": 211
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Turn right onto Candidstraße.",
            "verbal_transition_alert_instruction": "Turn right onto
Candidstraße.",
            "length": 0.933,
            "instruction": "Turn right onto Candidstraße.",
            "end_shape_index": 289,
            "type": 10,
            "time": 97,
            "verbal_post_transition_instruction": "Continue for 900 meters.",
            "street_names": [
              "Candidstraße"
            ],
            "begin_shape_index": 219
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Take the B 2R ramp on the
left.",
            "verbal_transition_alert_instruction": "Take the B 2R ramp on the
left.",
            "instruction": "Take the B 2R ramp on the left.",
            "end_shape_index": 297,
            "type": 19,
            "time": 9,
            "street_names": [
              "Candidstraße"
            ],
            "begin_shape_index": 289,
            "length": 0.124,
            "sign": {
              "exit_branch_elements": [
                {
                  "text": "B 2R"
                }
              ]
            }
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Merge onto B 2R.",
```

```
            "begin_street_names": [
              "B 2R",
              "Candidstraße"
            ],
            "verbal_post_transition_instruction": "Continue for 3 kilometers.",
            "instruction": "Merge onto B 2R.",
            "end_shape_index": 354,
            "type": 25,
            "time": 182,
            "street_names": [
              "B 2R"
            ],
            "length": 3.038,
            "begin_shape_index": 297
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Take the A 95 ramp on the right
toward Garmisch-Partenkirchen, Forstenried.",
            "verbal_transition_alert_instruction": "Take the A 95 ramp on the
right.",
            "instruction": "Take the A 95 ramp on the right toward Garmisch-
Partenkirchen/Forstenried/Fürstenried/Großhadern.",
            "end_shape_index": 360,
            "type": 18,
            "time": 14,
            "street_names": [
              "Heckenstallertunnel"
            ],
            "begin_shape_index": 354,
            "length": 0.305,
            "sign": {
              "exit_toward_elements": [
                {
                  "text": "Garmisch-Partenkirchen"
                },
                {
                  "text": "Forstenried"
                },
                {
                  "text": "Fürstenried"
                },
                {
                  "text": "Großhadern"
                }
              ],
              "exit_branch_elements": [
                {
                  "text": "A 95"
                }
              ]
            }
          },
          {
            "travel_type": "car",
            "verbal_pre_transition_instruction": "Keep right at the fork. Then
Bear right onto Luise-Kiesselbach-Platz.",
            "verbal_transition_alert_instruction": "Keep right at the fork.",
            "length": 0.025,
```

```
            "instruction": "Keep right at the fork.",
            "end_shape_index": 363,
            "type": 23,
            "time": 6,
            "verbal_multi_cue": true,
            "begin_shape_index": 360,
            "travel_mode": "drive"
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_multi_cue": true,
            "verbal_pre_transition_instruction": "Bear right onto Luise-
Kiesselbach-Platz. Then Take the ramp on the left toward A 95.",
            "verbal_transition_alert_instruction": "Bear right onto Luise-
Kiesselbach-Platz.",
            "length": 0.079,
            "instruction": "Bear right onto Luise-Kiesselbach-Platz.",
            "end_shape_index": 367,
            "type": 9,
            "time": 13,
            "verbal_post_transition_instruction": "Continue for 80 meters.",
            "street_names": [
              "Luise-Kiesselbach-Platz"
            ],
            "begin_shape_index": 363
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Take the ramp on the left toward
A 95. Then Keep left at the fork.",
            "verbal_transition_alert_instruction": "Take the ramp on the left
toward A 95.",
            "instruction": "Take the ramp on the left toward A 95.",
            "end_shape_index": 369,
            "type": 19,
            "time": 5,
            "verbal_multi_cue": true,
            "begin_shape_index": 367,
            "length": 0.037,
            "sign": {
              "exit_toward_elements": [
                {
                  "text": "A 95"
                }
              ]
            }
          },
          {
            "travel_type": "car",
            "verbal_pre_transition_instruction": "Keep left at the fork. Then Take
the B 2 ramp on the left.",
            "verbal_transition_alert_instruction": "Keep left at the fork.",
            "length": 0.087,
            "instruction": "Keep left at the fork.",
            "end_shape_index": 374,
            "type": 24,
            "time": 34,
            "verbal_multi_cue": true,
```

```
            "begin_shape_index": 369,
            "travel_mode": "drive"
          },
          {
            "travel_type": "car",
            "verbal_pre_transition_instruction": "Take the B 2 ramp on the left.",
            "verbal_transition_alert_instruction": "Take the B 2 ramp on the
left.",
            "instruction": "Take the B 2 ramp on the left.",
            "end_shape_index": 381,
            "type": 19,
            "time": 25,
            "begin_shape_index": 374,
            "length": 0.264,
            "sign": {
              "exit_branch_elements": [
                {
                  "text": "B 2"
                }
              ]
            },
            "travel_mode": "drive"
          },
          {
            "travel_type": "car",
            "verbal_pre_transition_instruction": "Merge onto A 95.",
            "verbal_post_transition_instruction": "Continue for 1.2 kilometers.",
            "instruction": "Merge onto A 95.",
            "end_shape_index": 387,
            "type": 25,
            "time": 81,
            "street_names": [
              "A 95"
            ],
            "length": 1.159,
            "begin_shape_index": 381,
            "travel_mode": "drive"
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Continue on B 2 for 80 meters.
Then Continue on A 95.",
            "verbal_transition_alert_instruction": "Continue on B 2.",
            "length": 0.078,
            "instruction": "Continue on B 2.",
            "end_shape_index": 388,
            "type": 8,
            "time": 3,
            "verbal_multi_cue": true,
            "street_names": [
              "B 2"
            ],
            "begin_shape_index": 387
          },
          {
            "travel_type": "car",
            "verbal_pre_transition_instruction": "Continue on A 95 for 11.5
kilometers.",
            "verbal_transition_alert_instruction": "Continue on A 95.",
```

```
            "length": 11.482,
            "instruction": "Continue on A 95.",
            "end_shape_index": 486,
            "type": 8,
            "time": 461,
            "street_names": [
              "A 95"
            ],
            "begin_shape_index": 388,
            "travel_mode": "drive"
          },
          {
            "travel_type": "car",
            "verbal_pre_transition_instruction": "Take exit 4 on the right onto A
9 52 toward Starnberg.",
            "verbal_transition_alert_instruction": "Take exit 4 on the right.",
            "instruction": "Take exit 4 on the right onto A 952 toward
Starnberg.",
            "end_shape_index": 510,
            "type": 20,
            "time": 51,
            "begin_shape_index": 486,
            "length": 1.172,
            "sign": {
              "exit_name_elements": [
                {
                  "text": "Dreieck Starnberg"
                }
              ],
              "exit_toward_elements": [
                {
                  "text": "Starnberg"
                }
              ],
              "exit_branch_elements": [
                {
                  "consecutive_count": 1,
                  "text": "A 952"
                }
              ],
              "exit_number_elements": [
                {
                  "text": "4"
                }
              ]
            },
            "travel_mode": "drive"
          },
          {
            "travel_type": "car",
            "verbal_pre_transition_instruction": "Merge onto A 9 52.",
            "verbal_post_transition_instruction": "Continue for 4 kilometers.",
            "instruction": "Merge onto A 952.",
            "end_shape_index": 556,
            "type": 25,
            "time": 150,
            "street_names": [
              "A 952"
            ],
            "length": 3.984,
```

```
            "begin_shape_index": 510,
            "travel_mode": "drive"
          },
          {
            "travel_type": "car",
            "verbal_pre_transition_instruction": "Continue on B 2 for a half
kilometer.",
            "verbal_transition_alert_instruction": "Continue on B 2.",
            "length": 0.458,
            "instruction": "Continue on B 2.",
            "end_shape_index": 563,
            "type": 8,
            "time": 28,
            "street_names": [
              "B 2"
            ],
            "begin_shape_index": 556,
            "travel_mode": "drive"
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Continue on Münchner Straße for
1.1 kilometers.",
            "begin_street_names": [
              "B 2",
              "Münchner Straße"
            ],
            "verbal_transition_alert_instruction": "Continue on Münchner Straße.",
            "length": 1.14,
            "instruction": "Continue on Münchner Straße.",
            "end_shape_index": 603,
            "type": 8,
            "time": 91,
            "street_names": [
              "Münchner Straße"
            ],
            "begin_shape_index": 563
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Continue on B 2 for 3.1
kilometers.",
            "begin_street_names": [
              "Hauptstraße",
              "B 2"
            ],
            "verbal_transition_alert_instruction": "Continue on B 2.",
            "length": 3.086,
            "instruction": "Continue on B 2.",
            "end_shape_index": 679,
            "type": 8,
            "time": 189,
            "street_names": [
              "B 2"
            ],
            "begin_shape_index": 603
          },
          {
```

```
            "travel_type": "car",
            "verbal_pre_transition_instruction": "Enter the roundabout and take
the 2nd exit.",
            "verbal_transition_alert_instruction": "Enter the roundabout and take
the 2nd exit.",
            "length": 0.083,
            "instruction": "Enter the roundabout and take the 2nd exit.",
            "end_shape_index": 689,
            "type": 26,
            "time": 4,
            "begin_shape_index": 679,
            "roundabout_exit_count": 2,
            "travel_mode": "drive"
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Exit the roundabout onto
Weilheimer Straße, B 2.",
            "begin_street_names": [
              "Weilheimer Straße",
              "B 2"
            ],
            "verbal_post_transition_instruction": "Continue on B 2 for 5.8
kilometers.",
            "instruction": "Exit the roundabout onto Weilheimer Straße/B 2.
Continue on B 2.",
            "end_shape_index": 800,
            "type": 27,
            "time": 242,
            "street_names": [
              "B 2"
            ],
            "length": 5.768,
            "begin_shape_index": 689
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Bear right onto Starnberger
Straße.",
            "verbal_transition_alert_instruction": "Bear right onto Starnberger
Straße.",
            "length": 0.237,
            "instruction": "Bear right onto Starnberger Straße.",
            "end_shape_index": 808,
            "type": 9,
            "time": 17,
            "verbal_post_transition_instruction": "Continue for 200 meters.",
            "street_names": [
              "Starnberger Straße"
            ],
            "begin_shape_index": 800
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_multi_cue": true,
            "verbal_pre_transition_instruction": "Bear left onto Feldafinger
Straße. Then Bear left to stay on Feldafinger Straße.",
```

```
            "verbal_transition_alert_instruction": "Bear left onto Feldafinger
Straße.",
            "length": 0.108,
            "instruction": "Bear left onto Feldafinger Straße.",
            "end_shape_index": 816,
            "type": 16,
            "time": 9,
            "verbal_post_transition_instruction": "Continue for 100 meters.",
            "street_names": [
              "Feldafinger Straße"
            ],
            "begin_shape_index": 808
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_multi_cue": true,
            "verbal_pre_transition_instruction": "Bear left to stay on Feldafinger
Straße. Then Bear right to stay on Feldafinger Straße.",
            "verbal_transition_alert_instruction": "Bear left to stay on
Feldafinger Straße.",
            "length": 0.101,
            "instruction": "Bear left to stay on Feldafinger Straße.",
            "end_shape_index": 820,
            "type": 16,
            "time": 9,
            "verbal_post_transition_instruction": "Continue for 100 meters.",
            "street_names": [
              "Feldafinger Straße"
            ],
            "begin_shape_index": 816
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "verbal_pre_transition_instruction": "Bear right to stay on
Feldafinger Straße.",
            "verbal_transition_alert_instruction": "Bear right to stay on
Feldafinger Straße.",
            "length": 0.313,
            "instruction": "Bear right to stay on Feldafinger Straße.",
            "end_shape_index": 830,
            "type": 9,
            "time": 21,
            "verbal_post_transition_instruction": "Continue for 300 meters.",
            "street_names": [
              "Feldafinger Straße"
            ],
            "begin_shape_index": 820
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "begin_shape_index": 830,
            "time": 51,
            "type": 8,
            "end_shape_index": 852,
            "instruction": "Continue.",
            "length": 0.78,
            "verbal_transition_alert_instruction": "Continue.",
```

```
            "time": 51,
            "type": 8,
            "end_shape_index": 852,
            "instruction": "Continue.",
            "length": 0.78,
            "verbal_transition_alert_instruction": "Continue.",
            "verbal_pre_transition_instruction": "Continue for 800 meters."
          },
          {
            "travel_type": "car",
            "travel_mode": "drive",
            "begin_shape_index": 852,
            "time": 0,
            "type": 5,
            "end_shape_index": 852,
            "instruction": "Your destination is on the right.",
            "length": 0,
            "verbal_transition_alert_instruction": "Your destination will be on
the right.",
            "verbal_pre_transition_instruction": "Your destination is on the
right."
          }
        ]
      }
    ],
    "summary": {
      "max_lon": 11.605507,
      "max_lat": 48.133167,
      "time": 2397,
      "length": 38.536,
      "min_lat": 47.943157,
      "min_lon": 11.260186
    },
    "locations": [
      {
        "original_index": 0,
        "type": "break",
        "lon": 11.6046,
        "lat": 48.133099,
        "side_of_street": "right"
      },
      {
        "original_index": 1,
        "type": "break",
        "lon": 11.2759,
        "lat": 47.941898,
        "side_of_street": "right"
      }
    ]
  }
}
```