St. Pölten University of Applied Sciences

/medien & digitale technologien

/fh/// st. pölten

Retrieval of multimodal location-based data from the web

Bachelor Thesis

Bachelor Course on Media Technology
at St. Pölten University of Applied Sciences

Submitted by:
Felix De Montis
mt171008

Advisor:
FH-Prof. Dipl.-Ing. Mag. Dr. Matthias Zeppelzauer

Vienna, 02.01.2021

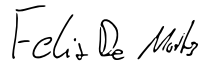
Declaration

- The attached research paper is my own, original work undertaken in partial fulfilment of my degree.

- I have made no use of sources, materials or assistance other than those which have been openly and fully acknowledged in the text. If any part of another person's work has been quoted, this either appears in inverted commas or (if beyond a few lines) is indented.

- Any direct quotation or source of ideas has been identified in the text by author, date, and page number(s) immediately after such an item, and full details are provided in a reference list at the end of the text.

- I understand that any breach of the fair practice regulations may result in a mark of zero for this research paper and that it could also involve other repercussions.



Date: 02.01.2021 Signature

Abstract

This bachelor thesis covers multimodal location-based datasets and how they can be gathered from different sources. The context around the topic is the scientific research in the field of real estate, in detail location assessment and analytics. The main part is about researching which kind of useful data modalities are available for datasets and how they can be used. In this process, a tool is developed to gather different types of data. They are made up of user generated social network posts, different satellite images and a mix of geological data. While creating the tool, problems which appear with different modalities are shown and solved. How a newly generated dataset can look like and how it compares to existing multimodal datasets is an important step which had to be considered. It has to be both consistent throughout executions of the tool but also set up in a way that both people and machines can work with it. As different modalities require different settings, everything can be set up using configuration files, which also allow generating a near identical dataset multiple times. To test the quality of both the tool and the generated datasets, different configurations are run to compare them with each other. Evaluating the tests shows that a flawless multimodal dataset requires both enough hardware resources and a satisfying configuration. Besides that, usage limits for some modalities have to be considered, depending on which modules are in use. It can be said that the tool does what it is intended for and that generated dataset can be used in further research.

Kurzfassung

Diese Bachelor Arbeit befasst sich mit multimodalen ortsbasierten Datensätzen und wie diese von verschiedenen Quellen gesammelt werden können. Der Kontext zu diesem Thema ist die wissenschaftliche Forschung im Gebiet der Immobilien, genauer gesagt der dazugehörigen Auswertung. Der Hauptteil behandelt die Recherche von nützlichen Daten, welche für neue Datensätze verfügbar wären und wie man diese verwenden könnte. Dabei wird ein Programm entwickelt, welches unterschiedliche Arten von Daten zusammenfasst. Die Liste beinhaltet nutzergenerierte Beiträge in Sozialen Netzwerken, unterschiedliche Satellitenbilder und mehrere geologische Daten. Während der Erstellung des Programms, sind bei den verschiedenen Arten an Daten Probleme aufgetaucht, welche aufgewiesen und gelöst wurden. Wie ein neu generierter Datensatz aussehen kann und wie dieser im Vergleich zu anderen multimodalen Datensätzen steht ist ein wichtiger Schritt, der nicht außer Acht gelassen werden kann. Er sollte sowohl zwischen Ausführungen des Tools gleichbleibend sein aber auch so aufgesetzt werden, sodass er für Mensch und Machine verständlich ist. Um unterschiedliche Arten an Daten zu bekommen, werden verschiedene Einstellungen benötigt, welche in eine Konfigurationsdatei gesetzt werden können. Diese ermöglichen das Erstellen von nahezu identischen Datensätzen. Um die Qualität des Programmes und der generierten Datensätze zu testen und auch diese daraufhin zu vergleichen, werden mehrere Durchgänge mit unterschiedlichen Konfigurationen ausgeführt. Die Auswertung der Tests zeigt, dass für fehlerlose Datensätze sowohl die passende Hardware zum Ausführen als auch eine zufriedenstellende Konfiguration benötigt wird. Noch dazu ist es wichtig, darauf zu achten, dass Verbrauchsgrenzen bei einigen Arten von Daten nicht überschritten werden. Es kann abschließend gesagt werden, dass das Tool das macht wofür es ausgelegt ist und dass die generierten Datensätze in weiterer Forschung verwendet werden kann.

Table of Contents

1 Introduction

In todays technological world, everything is dependent on data. No matter if you post something on a social media site or check your local weather, you either create or consume information in the form of digital data. Some people even go as far as calling data the new oil. (Bhageshpur 2019)

As with many aspects of our lives, scientific research is also profiting off the vast amount of readily available data. Location based data can be used to predict the weather, see environmental changes or detect regional trends.

The context in which this bachelor thesis will be written is the scientific research in the field of real estate, more specific location assessment. When assessing a specific location many factors can have different influences, some of those could be land use, temperature, air quality, accessibility or how trendy it is (Droj and Droj 2015). The data sources which will be accessed used in this thesis will be different satellite images, user generated social network posts and a mix of geological data.

A key problem for those assessments is the lack of a single multimodal dataset, containing data for a wide range of categories. When looking at specific categories, for example the weather, one can find datasets for the current weather, historical weather or maybe some fusions. When combining the need of weather data and some other category, the chances of finding a fitting dataset shrink. A single dataset covering a wide range use cases and categories doesn't exist until now.

The main goal of this Bachelor thesis is to create a tool, which can solve the stated problem. Besides that, insights into the creation of such tool will be gained. The tool will be programmed in python, a scripting language which is widely used in research. Its basic concept sees the tool taking in coordinates and returning a single structured dataset, which can be used in further steps for different research projects, especially the locations assessment.

Throughout the thesis, the term API will be used without further explanation. API is the abbreviation for "Application Program Interface" and is the point of entrance of web services, used to programatically access data. Some services offer APIs

openly and free of charge while others require authentication or even payments to gain access.

Which kind of data modalities are available and which are most useful for building a location-based multimedia dataset? What is the most efficient way to store location-based data? Will the tool work when creating large-scale datasets, more than thousand GPS locations? How can missing data be handled? Those questions will be answered in the fitting chapters.

The methodology of this bachelor thesis can be split up in 3 major sections: As the first part similar multimodal datasets, location-based APIs and services should be researched and evaluated for the usefulness. Those services will be implemented as proof-of-concepts to see if the data they return fits the requirements. Furthermore, the structure of the new multimodal dataset has to be thought of and how a newly developed tool can generate such datasets for various amounts of locations. Evaluation the quality of both the tool and generated datasets is an equally important step of the thesis.

The thesis has the following structure: Chapter 2 takes a look at similar datasets and location-based services. Chapter 3 shines light on the architecture of the tool, which services will work, what and how data might be scraped. Common problems and how the tool can be used will be brought up. The structure of the newly created dataset, how a single combined one can look like or how it compares to existing datasets will be picked up in chapter 4. Important key points are also how missing data will be treated or how things should be named. Chapter 5 deals with the evaluation of the newly created tool and its results, looking at scalability, availability and possible limitations. The final chapter (6) sums up the research and development done within this thesis.

2 Related datasets & services

When looking for related multimodal datasets or services, only few examples can be found. Many of those have a specific use case and thus consist only of data relevant to their use. Although the datasets are fitting for their use, they are irrelevant for the dataset, which will be created for this thesis.

Some datasets which stood out deal with social media data and autonomous driving:

ChrisisMMD (Alam, Ofli, and Imran 2018) looks at multimodal data of Twitter posts from natural disasters. It should help with the analysis of natural disasters done by humanitarian organisations. The data contains both text and multimedia content, both labeled regarding their relevance, which is a first of its kind. Another one, nuScenes by Caesar et.al, contains data from different sensors of autonomous vehicles. The multimodal dataset consists of 360-degree camera, lidar and radar data, all of those fully annotated. The dataset has 1000 scenes each with a length of 20 seconds using the described sensors (Caesar et al. 2020). The second dataset covering autonomous driving has its focus on off-road environments compared to urban ones like nuScenes. RELLIS-3D by Jiang et.al. contains not only 13,556 lidar scans and 6235 images, both of them fully annotated, but also raw data compromising camera images, lidar point clouds, stereo images, GPS and IMU data. Amongst others the dataset is intended for the development of advanced algorithms for navigation in off-road environments (Jiang et al. 2020).

All of the described multimodal datasets don't fit the needs of the dataset which will be used for location assessment. That's why it is also irrelevant, whether the datasets can be used or how they could be incorporated in a new diversified multimodal dataset.

Services in the form of APIs or smaller datasets which are multimodal don't exist or weren't found during research. As nearly all social networks have APIs nowadays and a vast number of open datasets with single modalities exist out there, a multimodal and broad ranging dataset can be created. Which of those are available will be discussed in Chapter 3.3.

3 Retrieval service

One prerequisite for both the tool and accompanied dataset is, that it works not only for some specific locations, but for every coordinate which gets passed to the tool. The main focus of the dataset and this paper lies in Austria, which is why some Austria specific datasets or APIs will be used. When using the tool with geographic locations outside of Austria, the gathered data will be smaller and its diminished.

Working with many different data services requires a unified and clear architecture. From time to time, services will cease to exist and new ones, covering interesting data modalities, will come up. That's why a consistent interface needs to be established.

3.1 Architecture and Usage

As already mentioned in the introduction, the tool will be written in Python. The file tree shown below gives an approximate overview, on how the source files of the tool will be structure.

```
├─ __init__.py
├─ modules/
│  ├─ satellite_google.py
│  ├─ streetview.py
│  ├─ instagram.py
│  ├─ elevation.py
│  └─ …
├─ helpers/
│  ├─ bounding_box.py
│  └─ …
└─ modules_data/
   ├─ elevation/
   └─ …
```

Figure 1. Structure of tool

Everything starts with `__init__.py` which can be included into any project. It exposes a single function which requires a list of geolocations and the path to a configuration file, latter will be described in the next sub-chapter. Although one wants to work with the gathered data, it wouldn't make sense to return all of the loaded data, as this would be probably very memory intensive. Instead, the function simply finishes, as soon as the dataset is created.

Besides the exposed function, `__init__.py` is also responsible for setting up the folder structure and preparing them for the different api modules. This has a simple and specific reason: the modules should be as clean and modular es possible and should not have to care about tasks, which matter all modules.

That is also the reason why each module, api or dataset, gets its own file, located in the `modules/` folder. A single function will be exposed to `__init__.py`, getting all the locations, configuration settings and authentication details from there. As this tool will be used in different projects, the required apis and datasets may differ. To save on resources, only necessary modules can be called, based on the program's configuration. To keep the modules from interfering with each other, data files, temporary files or files required for authentication can be saved in a corresponding folder inside of `modules_data`. An example for that is the elevation data for Austria, which is available as open data. The elevation is available as a two-gigabyte large GeoTIFF and it wouldn't make sense to bundle this large with the source code in case that it isn't needed. If the elevation data is required, the GeoTIFF will be downloaded by its module into the `modules_data/elevation` folder, if it isn't already available.

The whole tool works with geolocations, therefore it is important to use the same coordinate system throughout all modules to ensure compatibility. Some data sources come in specific coordinate systems specialised for Austria, other sources want pixel values to retrieve data. To help with this and also other issues, there are various helper functions in the `helpers/` folder. Chapter 3.4.1 goes more into detail about those problems and how they can be tackled.

3.2 Configuration

While the coordinates may differ for each execution of the tool, other parameters will probbably stay the same. That's why those settings are defined in a separate configuration file. An example for the configuration file could be following:

```
{
  "auth": {
    "google": {
      "key": "API-KEY",
      "secret": "API-SECRET"
    },
        "instagram": {
      "username": "ig_username_23",
      "password": "iG_Pa$$w0rD"
    },
    …
  },
  "modules": {
```

```
    "satellite_google": {
      "zoom": 12,
      "size": 600
    },
    "streetview": {
      "modes": [
        "equirectangular"
      ]
    },
    "satellite_bing": {
      "size": "2000
    },
    "elevation": {
      "grid": {
        "amount": 2,
        "distance": 50
      }
    },
    …
  },
  "save": {
    "root": "/home/felix/mmds/",
    "prefix": "testdata-"
  }
}
```

Listing 1. Example configuration file

To make the configuration file as explicit as possible, it is written in JSON, short for JavaScript Object Notation, as it can be used in python like a dictionary. The configuration can be split up in three different parts: authentication, module settings and saving.

The `auth` section is necessary for the modules where the API is not accessible unauthenticated. Some services require a combination of username and password, others want an "API-Key" and "API-Secret", others only a single "API-Key".

The modules which need to authenticate can access the corresponding authentication info from the configuration. That way code won't have to be modified, if different projects use different api secrets.

As the name already implies is the `modules` part responsible for defining both which modules will be used and how they work. Besides module specific settings it can be also specified wether data for a grid of geolocations should be gathered.

An example for a module specific setting is the `zoom`-level and `size` of Google Maps or Bing Maps satellite images which are necessary. The Google Streetview module has a `mode` setting, which states wether the result should be 6 different images, a cube map or a single equirectangular one.

The module independent `grid` setting defines if and how a grid around the requested point should be created. This can be especially interesting for elevation or landuse data. The configuration `"grid": {"amount": 2, "distance": 50}` says for example, that the grid should have **2** additional coordinates in each direction, with a distance of **50** meters between each point. That would result in a **5x5** grid of **200x200** meters. Difficulties which appear when calculating a grid of points with constant distance will be examined in chapter 3.4.1.

The last section of our configuration file, `save`, defines where a gathered dataset should be saved. Besides the absolute path to the parenting folder, a prefix can be also defined. Whenever the tool runs, it saves the new dataset under following location: `<root>/<prefix>-%Y-%m-%d_%H-%M-%S`. Chapter 4 goes into more detail, how the structure of the final dataset looks like.

3.3 Supported APIs and datasets

While researching for APIs and datasets relevant to the problem many interesting ones can be found. However, in the process of writing this thesis and while programming the tool, new some services were found whereas some of the initial ones turned out to be too complicated or useless.

To gather relevant services, those were collected in a spreadsheet with information about the data they offer, pricing, availability, authentication options, documentation urls and a category they loosely fit in. Following table shows a part of the spreadsheet, which is available with all columns in Appendix A.

Name	**Type**	**Pricing**	**Authentication**	**…**
Twitter	Service	Free but with limits	Yes, API keys, complicated developer account	…
Instagram	Service	Free	Yes, user account without 2FA	…
sensor.community	Service	Free	No	…
Wheelmap.org / accessibility.cloud	Service	Free	Yes, API keys	…
OpenStreetMap	Service	Free	No	…
…	…	…	…	…

Table 1. Portion of researched APIs and services

As the spreadsheet lists a large number of services and datasets, only a subset is tested or developed into modules. The services which are available within the tool are the following split up by category:

- Social Media
  - Twitter
  - Instagram
  - Flickr
- Environment
  - Weather (Weatherstack)
  - Elevation (Austria)
  - Land-use (Open Street Map)
  - Street size => Noise (Open Street Map)
  - Ambient Noise (Umgebungslärmkartierung)
- Images
  - Mapillary
  - Google Streetview
  - Google Maps Satellite
  - Bing Maps Satellite
  - Here Maps Satellite
  - Mapbox Satellite

3.3.1 Scraping modules

Unlike what was expected before starting this bachelor thesis, scraping content isn't required at all. The only service which isn't offering an official API is Instagram. However there exist various unofficial libraries for different programming languages, amongst them one for python which enables access to the private API used by Instagrams own iOS and Android Apps.

The problem which one faces when using the private API via the unofficial libraries will be exemplified in chapter 3.4.4.

3.3.2 Libraries

For some of the used APIs and datasets, there are python libraries available, to interact with the data. The libraries for Twitter and Flickr, use the official APIs, the one for Instagram is fittingly called `instagram_private_api`.

The data which comes from OpenStreetMap, street size and landuse, can be gathered via OpenStreetMaps own API, Overpass. A corresponding library which can translate the Overpass Query Language, the way to extract data, is also called overpass.

Besides the modules' specific libraries, various other ones are used to load text and images from the web, process this data or transform it into something else. Often used functionality like calculating the distance between two geolocations is extracted into the helper functions. How and which other libraries helped solve some problems, will be illustrated in chapter 3.4 below.

It is important, to always run the tool using the same libraries and their versions like when developing the tool. That is why all those libraries can be saved into a file. Typically, this file will be called `requirements.txt` and can be automatically generated by the python package manager using the `pip freeze > requirements.txt` command. When the tool gets used by someone else, they can then install all required libraries with `pip install -r requirements.txt`.

Besides libraries complicated functions which are necessary for multiple modules can be often found in blog posts or on the Questions-and-answer website "StackOverflow". Most of the researched library used to work with geolocations, don't offer the capability, to create an equally sized bounding box for each point on the globe. One "StackOverflow" answer [1] solves fortunately exactly this problem.

3.4 Common Problems and Limitations

While developing the tool, a number of problems, some more often than others, and limitations arose.

Following the Privacy scandals around Cambridge Analytica in 2018, various services stopped offering their data as easy as it was before the scandal. Twitter and Facebook restricted both what can be done with their APIs and who get's access to it (Perez 2018). In this process the API endpoints for Instagram and the one for images with tagged places were discontinued (Archibong 2018b, 2018a).

3.4.1 Different coordinate systems

If someone isn't working with geospatial data, it is probably unknown that there are multiple coordinate references systems, short crs. The most famous crs is the

[1] https://stackoverflow.com/a/238558/2046802

World Geodetic System WGS 84, also known as EPSG:4326. WGS 84 gets used by GPS and spans the whole world. Its bounds go from -180° to 180° as the longitude and -90° to 90° on the latitude. Another well-known crs used for Web Map Applications is the WGS 84 World Mercator projection, EPSG:3857. Compared to the first one, the Mercator projection only spans -85.06° to 85.06° on the latitude.

Coordinates are most often written in following two formats: Sexagesimal degrees which show degrees, minutes and seconds, 36°28'42.1"N 95°03'21.7"W, or decimal degrees, 36.478351°, -95.056035°, where the minutes and seconds get converted to the fraction of a degree. [2]

Most applications work with the coordinates in order of latitude and longitude in the format of decimal degrees. One issue which exists with some APIs is however that they accept the location with longitude before latitude. To circumvent wrong locations on another continent, latitude and longitude simply have to be flipped. Following figure visualises, where the bounds of Austria would lie, if the two values would be swapped.

Figure 2. Map showing bounding boxes for Austria and somewhere in Africa with flipped latitude and longitude

[2] https://en.wikipedia.org/wiki/Geographic_coordinate_conversion#Change_of_units_and_format

A problem which isn't so trivial to solve is that some datasets use different projections for their data. The elevation map for Austria is using the EPSG:31287 crs, called “MGI / Austria Lambert”. Unlike WGS 84, this crs uses meters as unit instead of degrees and spans 284970.54 to 575953.62 as latitude and 107724.11 to 680575.40 as longitude. The python library `pyproj`[3] can transform from one crs to another one, direction independent. That way WGS 84 coordinates can be used throughout the tool. As the elevation map comes in form of a GeoTIFF, other issues and how they were solved are described in chapter 3.4.7.

For datasets like land use or elevation, it is interesting to know how the data around the originally searched location looks like. Taking the elevation data as an example, one can see wether the location is on a slope or in the plain. The best way to get multiple data values around a single one is by creating a grid of `NxN` coordinates with the original one in the center. Depending on the requirements, the grid can consist of more or less coordinates with varying spacing.

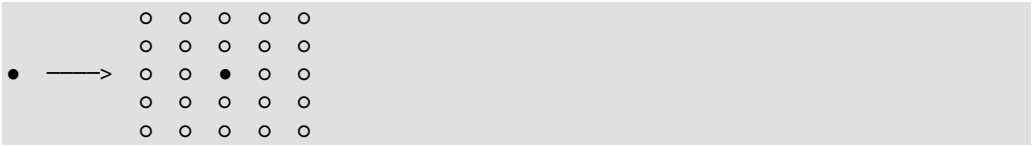


Figure 3. Process how a 5x5 grid gets generated out of a single point

Although this task looks simply, it sadly isn't. How how much space two longitudes have between each other depends on the latitudes. That is why one can't simply add a fraction of a degree to a WGS 84 coordinate, to move a specific distance. The distance between the points 47°N 15°E and 47°N 16°E, the latitude on which the City of Graz lies, is around 76km while 07°N 15°E and 0°N 16°E, directly on the equator, are approximately 111km apart from each other. To calculate how much the longitude has to be changed for a specific distance, the earth radius at the matching latitude has to be calculated.

3.4.2 Twitter

Twitter, which is mostly used to post near-real-time updates about happenings or events, has a very complicated relationship to their APIs. There are multiple API products, each with different costs and endpoints. The documentation is not very clear and if one figures out, how to use an api, many different limiting factors appear in a very late stage. Those are nowhere documented and the official documentation contradicts itself. Endpoints from newer API Versions (V2) which

[3] http://pyproj4.github.io/pyproj/stable/api/transformer.html#pyproj-transformer

replace the same functionality from older API Versions (V1.1) lack of key features like location filtering. Instead of planning to offer the same features, Twitter has no plan to change functionality except maybe in the future[4].

Besides the API limitations, the amount of available data is also a bit limited. According to a comment of a Twitter Developer Advocate, only 2% of users have geolocation enabled for their tweets[5]. When looking at the number of users globally, which were 330 million people as of the first quarter of 2019, approximately 6.6 million had locations for their tweets enabled (Twitter 2019). In Austria this number is already quite smaller, with around 160 thousand total users in 2019, only 3200 people had their locations enabled (artworx 2020).

3.4.3 Weatherstack

When looking for weather data, many services and providers can be found. However, only few offer both current and historic weather going further than a few days. Providers that offer data far into the past, cost a fortune most of the time, like "Weather API"[6] or "OpenWeather"[7].

The only provider which offers so called "Full Historical Data" at the cheapest price point is weatherstack which is also the reason why the tool uses this API to gather weather data.

While implementing the weatherstack API, various little quirks showed up. The first has to do with locations of weather stations and how they are not the same when requesting the weather for a place by name or by longitude and latitude pair.

To check wether this problem is only for one location or everywhere, I manually requested the weather for various locations. The city Steyr in Upper Austria serves as example: When looking for the weather in Steyr, the API returns a geolocation which is approximately the centre of the bounding box of Steyr. However, when requesting the weather for the exact location which came back from the previous request, the API now returns the place Steinersdorf, which is about 6km away from the centre of Steyr. A second location I used to try the API is Bad Schallerbach, also located in Upper Austria. When going through the same evaluation steps, the weather for a place 40km away gets returned. To rule out, that this problem only

[4] https://twittercommunity.com/t/v2-filter-search-by-location/144401

[5] https://twittercommunity.com/t/adding-rules-to-fetch-the-geo-location/145359/3

[6] https://www.weatherapi.com/pricing.aspx

[7] https://openweathermap.org/full-price#history

exists in Upper Austria, I also tried locations in Vienna and Lower Austria. When asking the weatherstack support about this inconsistency, they say that they rely on third parties for the wether stations and will look into the problem of them being up to 40km away from the requested location.

Another annoyance is that the historical data API by default only allows time series up to 60 days when using a start and end date. This can be circumvented by using a different endpoint which is used to get the historical weather for specific days. When combining this with the fact, that multiple queries can be combined using semicolons, custom time series can be created by the tool[8] and then passed as different days.

Although weatherstack is one of the cheapest weather APIs, there could be still limitations when using the lowest pricing level. The lowest level is limited to 50.000 requests per month, however, each day of a time series for historical data counts as a request. As the tool will be tested with around 1000 locations, the town halls through Austria, it would be possible to get only historic data for 50 days for each location. To test and develop this bachelor thesis, a sponsored Business account is used, which allows up to 1 million requests[9].

But as mentioned already in the beginning of this subchapter, other providers offering the same capabilities as the starter level cost as much as the business level or even more.

3.4.4 Instagram

Compared to all other services and datasets, Instagram is the only one in use which isn't offering a public API. Only Facebook Business Partners get access to a special API, to offer marketing tools to companies which require such tools.

A solution to this problem is the use of Instagrams internal private API. There are two difference APIs, the one used by the Web APP, supports most features but not all of the ones available in the native APPs. The native APPs use a different API Endpoint which offers all additional features like posting of media, stories and more.

Luckily there are libraries implementing those private APIs in many programming languages, among them also for python. They make the usage of the APIs as easy as official libraries for other services, however as they are actually not certified or

[8] https://docs.python.org/3/library/datetime.html#timedelta-objects
[9] https://weatherstack.com/product

allowed by Instagram, the risk of them breaking is high. Another issue which could arise is that the account could get blocked for misuse. In that case it would be useful to create a new Instagram account just for the API, to prevent a blockage on a personal account.

When looking at data itself, which can be gathered by the tool, other small challenges appear. Images can't be directly tagged to a geolocation but only to a point-of-interest, POI, which is supplied by Facebook. Therefore, when looking for matching images for a location, it can be only search for POIs near the requested location and the images which were taken there. If a POI is very popular or famous, the images could be only of one specific kind.

If a location is a selfie hotspot, most images will be of people. Besides famous building those locations could be also fitness studios, where everyone takes selfies after their workout sessions. Images tagged at the Leopold museum are mainly from the exhibits which are available there. Bars and Restaurants, for example Figar in the 7th Viennese district, show mostly images of their dishes. This has to be considered when evaluation the gathered images.

3.4.5 WMS

Noise Data from the ambient noise field mapping comes in the OpenGIS® Web Map Service Interface Standard[10], short WMS, which is a specification which defines types of available Map Data and how to retrieve it.

Compared to most other APIs, WMSs main purpose is serving images as layers on top of other maps. In the context of our Tool, we aren't interested in large sections of a map but in one or more specific points. The standard supports the retrieval of single point, but as this isn't the focus[11], finding out about it isn't that easy[12].

[10] https://www.ogc.org/standards/wms

[11] https://en.wikipedia.org/wiki/Web_Map_Service

[12] https://docs.geoserver.org/stable/en/user/services/wms/reference.html

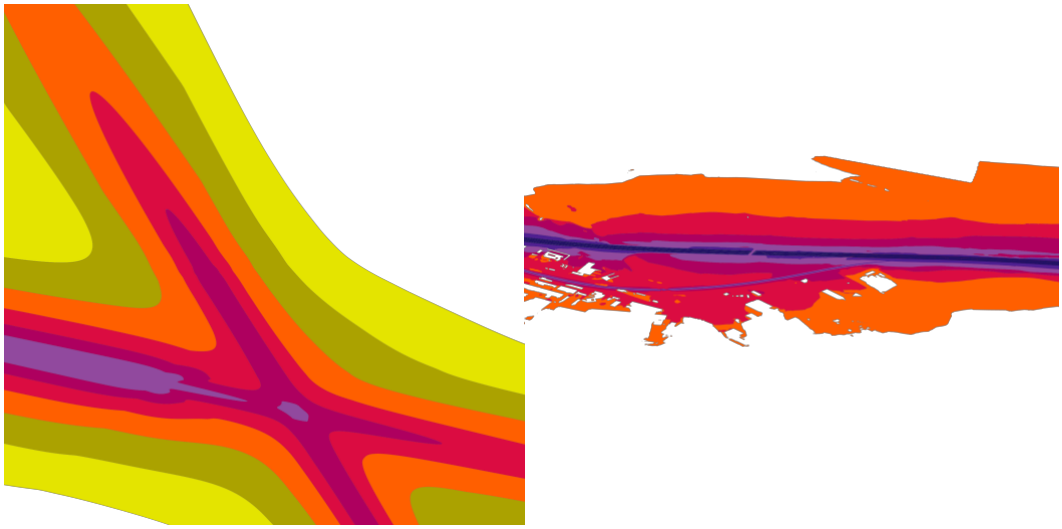


Figure 4. Layer requested using WMS showing the noise level of one of the runways of VIE (left) and a street located nearby (right)

Wether multiple layers can be requested at once depends also on the implementation of WMS. The service offering the ambient noise data returns errors when asking for the data of different layers although images are supported. When contacting the authority in charge of this service, they can't explain the problem and stated, that they will investigate the issue. To solve this problem, every layer has to be requested on their own.

3.4.6 OpenStreetMap

OpenStreetMap, widely known as OSM, is an open-source alternative to Google Maps, where all of the data is user contributed. Different kind of data is available to extract, from point-of-interests to street information or public transportation routes. The tool uses OpenStreetMap to gather land-use and street speed limits. The street speed limits can used to approximate the noise levels of streets.

The data available in OpenStreetMap can be access over the own Overpass API. It interprets an own syntax called Overpass Query Language, to search for different types of features, combine results and filter those. Depending on the requirements only simple information about locations or fully fletched shapes for buildings and regions can be extracted.

A problem which appears when using the Overpass API is that searches can be done within a bounding box or a radius around a location. If the bounding box is too small, polygons for specific features won't be loaded. That's the reason why looking at the land-use isn't possible using a bounding box only somewhat bigger

than a specific location. The figure shows a comparison of retuned data based on bounding box size.

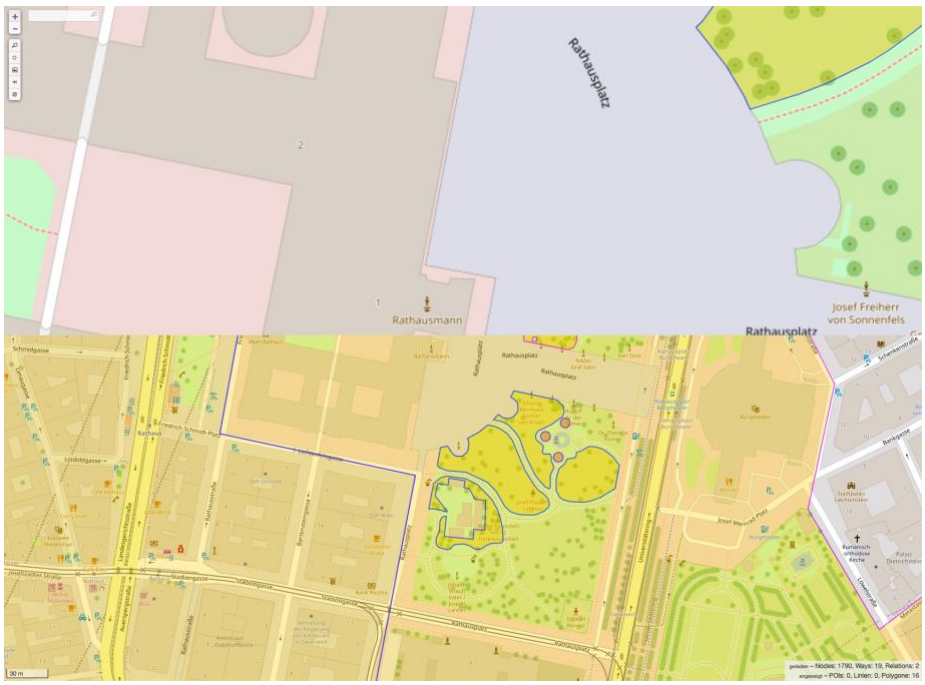


Figure 5. Top half shows results with small bounding box[13]
Bottom half shows results with large bounding box[14]

While the large bounding box solves the problem of not getting the data, it brings its own issues. It is not possible to get the value of a feature at a specific location, that is why all returned polygons have to be checked if the requested location is within a polygon. OpenStreetMap supports besides normal polygon made out of geolocations also special multipolygons for OSM Relations. They are made up of outer polygons and inner polygons, which cut away parts.[15]

"Shapely", a python library for geometry, helps with both normal polygons and multipolygons as it can treat polygons as shells and corresponding holes. It can also tell, wether a specific location is within a multipolygon to determine quickly, which land-use polygon encloses those points.[16]

[13] https://overpass-turbo.eu/s/10ZB

[14] https://overpass.turbo.eu/s/10ZA

[15] https://wiki.openstreetmap.org/wiki/Relation:multipolygon

[16] https://shapely.readthedocs.io/en/stable/manual.html#polygons

3.4.7 GeoTIFF

GeoTIFF is a format which saves its data in the value of pixels. The TIF format being a lossless format enables it to save data with high accuracy and allows all compression algorithms which work with normal TIF images. As images are made up out of pixels, the type of projections and location has to be saved in the GeoTIFFs metadata.

As already described in the previous sub-chapter about different coordinate systems, the locations in WGS 84 have to be translated into the projection used by the GeoTIFF. To get the value for a specific location, the projected geo location has to be converted into the corresponding row and column of the pixel as GeoTIFFs are available in different resolutions. The python library "rasterio" can do the translation to pixel coordinates and extraction of values.

One problem which appeared with the GeoTIFF for Austrian elevation is that the image has a size of 58061 by 31793 pixels resulting from the approximate width of 580km and height of 318km with a resolution of 10 meters. Such big images come also with large file sizes, the elevation TIF has nearly 2GB. Therefore, it isn't possible to requests the file for each location like when using WMS or other APIs. The file has to be downloaded in advance or on first use and can then be loaded to get location data.

Although memory isn't such a problem nowadays anymore, it should be noted, that loading the elevation GeoTIFF requires around 7.5GB of RAM. Wether this is an issue with the specific file used, the python library which loads the TIF or something else, couldn't be determined.

3.4.8 Satellite Images

Like weather data, there are many sources and providers for satellite images. The Sentinel satellites from ESAs Copernicus program capture various environmental information about our earth. The Sentinel-2 satellite pair takes multispectral images of the earths surface in a resolution of 10 to 20 meters. What makes the Sentinel satellites an interesting contender for the images is that the data is available for free, however it is really difficult to get image data for a specific geolocation.

As the available images have the shape of the orbits the satellites take, one has to find out in which images the location could be located. When matching images are found, it still has to be checked, if the images contain data for the exact location. If only the near surrounding of a location is interesting, this would have to be extracted from the larger image.

To circumvent all those complicated steps, it is faster and more reliable to get the images from maps providers. Different providers offer different quality of images for different locations. To get good results multiple services will be used for the satellite images. As mentioned in the previous chapter 3.3 the following services will be the source: Google Maps Satellite, Bing Maps Satellite, Here Maps Satellite and Mapbox Satellite. All of them cost something but come with free usage quotas which will be enough for the tool.

The resolution of the available images isn't as good as the Sentinel ones but the approach of using multiple sources should handle that issue. Besides the resolution of the image, the size of those images is also limited with Google Maps having the smallest ones. It is however possible to request images twice the size when contacting their support[17].

Another issue which exists with every map provider is the available zoom level. Different locations allow different zoom levels depending on the resolution of the image data to which they have access to, that way one can zoom for example further into San Francisco than Vienna. If the image should be with the maximal zoom level, one is out of luck, as it is not possible to get that information. Google Maps show an error message, if an image is generated with a zoom level too high. Although the issue of not being able to get the maximum zoom level is known already since many years, it is still an open one[18]. Most Map providers offer therefore an approximation on what can be seen at which zoom level.

[17] https://developers.google.com/maps/documentation/maps-static/start#Largerimagesizes

[18] https://issuetracker.google.com/issues/35821504

Zoom Level	Google	Mapbox	OpenStreetMap
0/1	World	The Earth	whole world
3		A continent	largest country
4		Large islands	
5	Landmass/continent		large African country
6		Large rivers	large European country
10	City	Large roads	metropolitan area
15	Streets	Buildings	small road
20	Buildings		A mid-sized building

Table 2. Comparison of zoom levels form different map providers

Mapbox, OpenStreetMap and Here Maps offer also approximations or functions to calculate the meters per pixel based on zoom level and latitude[19] [20] [21].

[19] https://docs.mapbox.com/help/glossary/zoom-level/#zoom-levels-and-geographical-distance

[20] https://wiki.openstreetmap.org/wiki/Zoom_levels

[21] https://blogs.bing.com/maps/2006/02/25/map-control-zoom-levels-gt-resolution/

4 Dataset structure

Retrieving the data for different locations is only part of the tools use case. To make use of the gathered data it has to be in a structured and logical format. If the tool is run multiple times with the same or different configuration, the resulting dataset should not change much.

4.1 How could one combined dataset look like

To find out how such a dataset could look like it is important to know what will be saved. Different modules have different outputs or combinations of them. Nearly all of the modules return some kind of machine-readable data. This original can appear as lists, dictionaries or simple values. Some modules return also urls to media besides metadata, which can be also downloaded.

If modules are configured in a way, that not only a specific location but a grid of neighbouring geo locations will be queried, both the grid and its values have to represented somehow. This is also the case if there is no data available for a specific location and only for point-of-interests nearby. As there can be multiple POIs, the results of those shouldn't be combined.

When creating a large dataset, various modules request data for many locations. To avoid confusion, the dataset has to be well structured and easily readable for both humans and machines, as in most cases there will be done further processing on the data. A way to keep datasets as consistent as possible is to use the same file names for each module. The final data should be saved in one machine readable json file. In the case that the original data is also interesting in a later step, this will be also saved. If data can't be requested or if other errors appear, those should be also saved, to make it easier in later steps to distinguish easily between correct and faulty datasets.

4.2 Existing data model (related works)

When looking at the related multimodal datasets, nuScenes is the largest one but unlike this new dataset, nuScenes is made up of linked data. The new dataset contains mainly data from different sources which have little to no connections with

each other. This is also the reason why the structure of nuScenes isn't relevant. The only connecting data from the various modules is the common location.

4.3 Proposed data model

The root of each generated dataset, the location where everything will be saved, is defined in the configuration file which was described in chapter 3.2. The `save` section defines both the parent folder and a prefix, the latter one will be prepended to each generated folder.

```
"save": {
  "root": "/home/felix/mmds/",
  "prefix": "testdata-"
}
```

Listing 2. Example for save section of configuration file

The example config above declares `/home/felix/mmds` as the parent folder, the prefix for each dataset is `testdata-`. Combined with the date and time from when the tool gets executed the path to the first level will result in something like this: `/home/felix/mmds/testdata-%Y-%m-%d_%H-%M-%S`. Having both date and time in the folder name assures that the tool will never override a previous execution, which could result in data loss.

There are two possibilities, how the dataset itself could be structured:

- Locations as the second folder level wrapping each module and its data
- Modules as second folder level wrapping each location and the gathered data.

The first approach seems more logical, as it is easier that way to get all the data for a single location. The second approach could be useful if only satellite images for all locations are required. However, this could be also achieved if the tool is run having only the satellite images configured.

Because of the reasons above the folder for each execution are folders named after the requested location in the format of `<latitude>_<longitude>` with the coordinates as decimal degrees. Some locations might gather data for multiple adjacent points, this will be saved within the module to keep second folder level clean off the additional geo locations.

How the data looks inside of each module depends on the type of data returned by the module. If some APIs return JSON, it get's saved as `response.json`. Information which gets generated from the tool or converted out of the response

will be saved as `data.json`. Images which are part of the response or data will be saved under a subfolder called `images`.

Besides the machine-readable information both `debug.log` and `time.log` files get saved in the root of the dataset. The first file contains the logging output from all modules, each line prefixed with a modules name. It can be used to easily find errors when data is missing in the final dataset. `time.log` saves the start and finish time for each module and also for the whole dataset. If execution times seam flawed, those times can be checked to see wether a specific module was too fast because of an error.

A possible folder structure with the Google Satellite, Streetview, Instagram and Mapillary modules configured, could look something like Figure 6. It shows also the peculiarity of the satellite, Mapillary and Instagram modules.

As there is no additional data to the satellite images, it wouldn't make sense to save some arbitrary information inside of a `data.json` file. On the other hand, Instagram has multiple layers of data, a single posting can be either a single image or video, or a collection of those as a slider. If a posting is such a collection, the content will be saved within an extra folder matching the postings id.

Mapillary is also a module with a more complicated data tree. The images one gets from this service can be 360° ones, however most of the are regular two-dimensional photos. As a consequence, it can't be sure wether a requested location is visible inside of an image. To mitigate this issue not only the nearest images will be saved, but also other images from sequences they are part of. How many of the previous and next images will be saved as a set can be configured. The module saves all matching sequences as folders and the associated images within them.

```
mmds/
├─ testdata-2020-10-07_08-10-50/
│   ├─ 41.303921_-81.901693/
│   │   ├─ sattellites_google/
│   │   │   └─ satellite.png
│   │   ├─ streetview/
│   │   │   ├─ data.json
│   │   │   └─ images/
│   │   │      ├─ equirectangular.jpg
│   │   │      ├─ part1.jpg
│   │   │      ├─ part2.jpg
│   │   │      └─ ...
│   │   ├─ instagram/
│   │   │   ├─ data.json
│   │   │   ├─ response.json
│   │   │   └─ images/
│   │   │      ├─ p42342pbaw52s3bn.jpg
│   │   │      ├─ a898av34dg25aaw8.jpg
```

```
│   │   │       ├─ da656wa6d57aw5d6/
│   │   │       │   ├─ da656wa6d81gw3m7.jpg
│   │   │       │   └─ da656wa6d61oy5d8.jpg
│   │   │       └─ ...
│   │   └─ mapillary/
│   │       ├─ data.json
│   │       ├─ images.json
│   │       └─ images/
│   │           ├─ bu7DVLfq3zL5tsXz_TN4Vw/
│   │           │   ├─ 9DaQQnjC2dAT9Mt2RoTxTg.jpg
│   │           │   ├─ DII7ubMJMU76CtEN7m433g.jpg
│   │           │   └─ ...
│   │           ├─ B7tuE2dFVC6oYF8yO7cgHw/
│   │           │   └─ ...
│   │           ├─ B2wCmCIEiEPj4bvI2fUK0Q/
│   │           │   └─ ...
│   │           └─ ...
│   ├─ 48.212562_15.630233/
│   └─ 48.200709_16.352543/
└─ testdata-2020-11-02_14-02-32/
```

Figure 6. Possible folder structure of a single generated dataset

4.4 Handling of missing data

A generated dataset is not always perfect. Often some data points are missing because of various reasons, those could be either related to the requested location, type of data or simply error within the generating tool.

Missing data gets most often defined using the following three categories (Karkare 2019):

- Missing completely at Random (MCAR)
- Missing at Random (MAR)
- Missing Not at Random (MNAR)

Missing completely at Random means, that there is not relation between the missing data point and other parts of the dataset. This can happen when APIs are not reachable or show some kind of error.

When talking about **Missing at Random** the missing data points are not missing because of themselves but show a relation to the rest of the dataset. Locations in more rural areas might have a lower density of data when requesting social media data.

Missing Not at Random describes the issue when data points are missing because of the data itself. This could happen with the elevation data, when a dataset gets generated for a location outside of Austria, as the original data exist only within Austria.

There are two ways how some kinds of missing data could be mitigated: refining search queries before running the tool or interpolating data points from similar data.

Many modules look for data within a bounding box or circle with a given radius. Instagram and Twitter allow searching for words too. If previous runs of the tool show no or little results for some locations, it can already help to increase the radius. Looking for posts on Instagram and Twitter without words in the search can also improve the number of results. It is however important to remember, that larger search queries can return more imprecise information.

When looking at the elevation or land use data, missing points can be maybe interpolated from locations nearby. Both modules support the tool's grid option to retrieve multiple values for locations nearby, that way interpolating the elevation is relatively easy. Having a 3 by 3 grid with the centre one missing, one can simply sum the remaining 8 values up and divide the result by 8.

15	20	25
20	?	30
25	30	35

→

15	20	25
20	**25**	30
25	30	35

Table 3. Process showing how a value can be interpolated from a 3x3 grid

The module requesting speed limits for neighbouring streets exists as a fallback when ambient noise data isn't available for a specific location. As most of the noise gets generated from traffic, the speed limits and lane count can be used to estimate the noise generated on the streets.

The next chapter Evaluates some generated datasets and sheds also a light on the modules which produce missing data and how the issues can be possibly solved.

5 Evaluation of retrieval service

To evaluate the efficiency and quality of the tool, it has to be tested with various locations in various regions. Both the number of locations and the amount of available data at specific location can change, how the tool performs.

5.1 Scalability

To test the scalability of the tool, it is important to have many different locations not too close to each other to get a diverse dataset. Luckily OpenStreetMap allows easy ways to retrieve lists of locations of various sizes all within Austria. Different tests are made using manually selected locations while development, smaller lists while testing for bugs and bigger ones for final evaluation. Following table shows the sample sizes:

Manually	Thalia	Müller	Merkur	McDonald's	Town halls
2-10	~30	~50	~110	~180	~1700

Table 4. Types of location and corresponding sample sizes

The first tests running the tool only with few locations showed no problems. The next test with McDonald's locations showed issues with the land use module, which didn't appear when using less locations. Besides that, multiple requests failed to connect to a server. To check wether these issues also appear with other locations and to fix the issues, the same configuration was run with all Thalia locations, as fewer locations mean faster run time.

The issue with land use was, that the OpenStreetMap multipolygons, already mentioned in chapter 3.4.6, can be made up by multiple outer polygons who don't necessarily have to touch each other. Shapely, the library in use, can only work with a single outer polygon and multiple inner polygons as holes. Besides that, sometimes multipolygons are defined by a number of open ways, a series of locations. When such a series only contains two locations, Shapely isn't possible to transform it to a polygon.

If the tool makes more then approximately 30 requests per seconds, various APIs and services report, that they can't reach their endpoints because of the errors like the following from the Flickr module:

```
requests.exceptions.ConnectionError:
HTTPSConnectionPool(host='api.flickr.com', port=443): Max retries exceeded
with url: /services/rest/ (Caused by
NewConnectionError('<urllib3.connection.HTTPSConnection object at
0x2d788f0a0>: Failed to establish a new connection: [Errno 8] nodename nor
servname provided, or not known'))
```

Listing 3. Error messages from Flickr module not being able to reach the server

When taking a closer look, the last part of the error message suggests, that while creating the connection to the server, the server couldn't be found. This might has to do with the sheer amount of requests or the internet connection which was used for part of the tests. To eliminate the second possibility, some tests are also done on a faster network.

Further investigation shows that the internet speed isn't the problem but as the tool can run multiple modules in parallel, opening so many connections is too much for the machine running or python. This can only be fixed by running all modules and all locations in series, which increases the duration quite a bit but results in return with a complete dataset.

With this bug fixed, anew tests using the locations from Merkur were run. One run where all modules succeeded with all locations took about 90 minutes, dividing this duration by the 110 locations yield a time of 50 seconds per locations.

Before running the test for the approximate 1720 locations for town halls, estimating the duration results in nearly 24 hours. As this would be too long for a single run, only half of the locations with an estimated 12 hours were run. To keep the time even lower, a configuration was used which downloads less images. The tool finished after 400 minutes or 6 hours 40 minutes. One problem which appeared thereby is that both Twitter and Instagram limit the number of requests within a specific time frame.

Twitters API is public, therefore the rate limiting of 450 searches per 15 minutes can be found in their docs[22]. When more than 450 searches are made, the time until requests are allowed again gets sent in the header. As Instagrams API is not open, it is only possible to guess the amount of requests, before rate limiting kicks in. When looking at the log produced by the tool, the first 433 searches worked. Both rate limiting issues can be solved at the same time by generating only datasets with a maximum amount of 430 locations.

[22] https://developer.twitter.com/en/docs/twitter-api/rate-limits

Instead of running into the same issues with the second half of the town hall locations, only 430 locations were used, a quarter of all locations. Besides Instagram, everything ran flawless, this module failed after one search with the rate limiting error messages. It seemed like the account was blocked because of the excessive usage. Running the same locations only with the Instagram module enabled and a new account worked without a problem. Summing the times running all modules and running only Instagram equates to 3 hours 50 minutes. Dividing this by 430 locations shows that each location took around 32 seconds.

The last quarter containing 435 locations also had problems with Instagram but this time there was no rate limiting but a privacy consent popup, which had to be clicked in the app. This could have happened, as the account was newly created. The total summed up time is 4 hours 9 minutes which means 34.5 seconds per location.

Locations	Total w/ faulty IG	Total w/o IG	IG
430	13376s	13239s	547s
435	14291s	14115	801s

Table 5. Snapshot of speed diferences between execution with 430 and 435 locations

Although there are only 5 more locations, comparing the times between the third and fourth quarter shows that nearly all modules were slower when running the fourth quarter, accumulating a difference of 18 minutes and 50 seconds. The most possible reason is that the internet speed was a bit slower, but a slowdown of only 8% carries no weight.

Module Name	430	435	Diff
satellites_google	620	625	5
streetview	1.693	2.009	316
satellites_bing	281	328	47
instagram	137	176	39
weatherstack	313	386	73
twitter	207	249	43
satellites_here	362	421	59
satellites_mapb	538	603	65
elevation	79	71	-8
landuse	3.054	3.112	57
mapillary	1.326	1.375	49
flickr	1.241	1.355	114
streetsize	1.292	673	-619
noise	2.233	2.908	675
final time	13.377	14.292	915

Table 6. Detailed comparison between execution of 430 locations compared to 435 locations, values are given in seconds

5.1.1 One location vs. 100 locations vs. 1000 locations

The main difference between running the tool with one, 100 or 1000 locations is the speed of execution. As stated above, both Twitter and Instagram are rate limited, which indirectly forces a limit of 430 locations per run. When running the tool without those two modules, the 860 locations also succeeded. It can be assumed, that therefore also 1000 locations will work but because of resource limits, this wasn't possible to test.

Finding one to ten good locations to test the tool is easy, as different locations return different results. However, getting 100 up to 1000 locations is quite a challenge. An easy way to get different locations of a single kind is using OpenStreetMaps Overpass API, which is also used by the land use and streetsize modules.

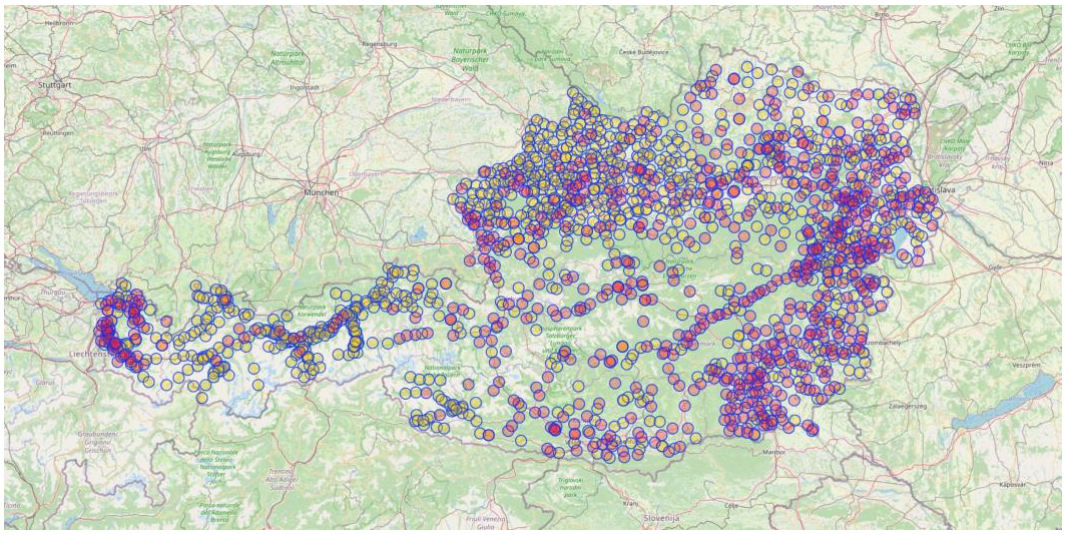


Figure 7. All town halls in austria as of 05.12.2020[23]

To get all town halls in Austria with their location, name and other meta information, one can query the Overpass API for town halls which can be found by their tag `amenity=townhall`[24]. As of 05.12.2020 there are 1853 locations with this tag, when filtered for duplications and POIs which are not town halls, around 1720 remain, which were used in the tests.

The other locations for Thalia, Müller, Merkur and McDonald's can be found in OpenStreetMap with the `brand=` tag. A nice thing about those stores is, that all of them have multiple subsidiaries all over Austria.

5.2 Availability

As already mentioned in the begingin of chapter 3, the focus of this paper lies on Austria. Nevertheless, many of the used APIs work in Europe or even Globally. Some of the datasets and services from bigger companies or projects, like Google or Mapillary, even work better in Countries like the United States, as they have their origin there. The elevation or noise datasets are provided by the Austrian government and thus cover only Austria.

[23] https://overpass-turbo.eu/s/10iC

[24] https://wiki.openstreetmap.org/wiki/Tag:amenity=townhall

For something like landuse, there were multiple datasets and services with different regional focus, but OpenStreetMap was the easiest to work with and therefore used in the context of this project.

Generally, one can say, that the availability is a double-edged sword. More data is available in locations with higher population density, especially when looking at social networks. Taking Instagram as an example, much more location tags are available in a city compared to the countryside, this leads to the situation, that the nearest location in a city might actually not be what is wanted but in low-density areas the correct one is the only one. The same phenomenon can be seen with the 360° Streetview images. As with Instagram, cities are well covered in comparison the roads on the countryside. However, when looking at Streetview images for McDonald's Restaurants, the ones on the countryside often show the correct building while McDonald's inside of shopping malls aren't that often visible as the Streetview images are taken from the street or maybe some different store.

When adding a search word to the queries for Twitter and Instagram, the quality of results would be much higher, the number of posts however less. Looking for Tweets containing the phrase "Gemeindeamt" while running all town hall locations results in 0 tweets, as nobody tweets using this word. But in comparison, nearly all locations found on Instagram are the correct town halls.

6 Conclusion

Although some might think that scraping data and creating a new multimodal dataset is an easy task, there are many intricacies both throughout the the modules implemented for the tool and when running the tool. Different types of data gets offered in various ways, either as standalone downloadable datasets, through special APIs or as part of a larger collection.

When comparing the structure and use case of the newly generated multimodal datasets to other multimodal datasets, all of them are smaller in size or handle only one kind of data.

The high number of subchapters below chapter 3.4 shows that nearly all data sources have their quirks, which have to be handled. This starts with the way how one interacts with the API endpoints; some APIs are well documented and allow easy use through third party libraries. Other sources are less self-explanatory and need deeper research and experimenting to make them work, like the noise data which is based on WMS. The fact that some sources use unconventional coordinate systems and projections should't be left out of consideration.

It is very important to have a well-structured dataset which keeps both consistent when using different configurations or when using changing locations. A key feature of a dataset has to be reusability in further steps, so everything which gets generated has to be machine readable. The dataset may not be a black box and therefore everything is logged to understand why a dataset is the way it is.

As mentioned in the previous chapter, both scalability and availability have and impact on how the final dataset looks like and what data it contains. To receive a final single dataset which matches the expectations depends on both the locations given to the tool and how the configuration is made up. Configurations with higher precision result in more accurate datasets, however the this can also backfire. Having a configuration too detailed might make the dataset unusable.

Scraping and loading data is both very memory and storage demanding and therefore it is important to use hardware which is strong enough to handle the task. A thing which can't be ignored is the big amount of external data which requires also a well functioning network. If a spotty wifi connection is used, this might result in dropped connections and thus an incomplete dataset.

Conclusion

Having such multimodal datasets and being able to generate them for any location in a matter of few hours enable new ways to analyse and work with location data. Besides location assessment for real estate, which was the initiator for this thesis and tool, many other questions in this area can be dealt with.

Bibliography

Alam, Firoj, Ferda Ofli, and Muhammad Imran. 2018. “CrisisMMD: Multimodal Twitter Datasets from Natural Disasters.” *ArXiv:1805.00713 [Cs]*.

Archibong, Ime. 2018a. “A Platform Update.” *About Facebook*. Retrieved December 16, 2020 (https://about.fb.com/news/2018/07/a-platform-update/).

Archibong, Ime. 2018b. “New Facebook Platform Product Changes and Policy Updates.” *Facebook for Developers*. Retrieved December 16, 2020 (https://developers.facebook.com/blog/post/2018/04/24/new-facebook-platform-product-changes-policy-updates/).

artworx. 2020. “Österreich - Twitter-Nutzer 2019.” *Statista*. Retrieved December 14, 2020 (https://de-statista-com.ezproxy.fhstp.ac.at:2443/statistik/daten/studie/296135/umfrage/twitter-nutzer-in-oesterreich/).

Bhageshpur, Kiran. 2019. “Council Post: Data Is The New Oil -- And That’s A Good Thing.” *Forbes*. Retrieved November 6, 2020 (https://www.forbes.com/sites/forbestechcouncil/2019/11/15/data-is-the-new-oil-and-thats-a-good-thing/).

Caesar, Holger, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. “NuScenes: A Multimodal Dataset for Autonomous Driving.” Pp. 11621–31 in.

Droj, Laurenţiu, and Gabriela Droj. 2015. “Usage of Location Analysis Software in the Evaluation of Commercial Real Estate Properties.” *Procedia Economics and Finance* 32:826–32. doi: 10.1016/S2212-5671(15)01525-7.

Jiang, Peng, Philip Osteen, Maggie Wigness, and Srikanth Saripalli. 2020. “RELLIS-3D Dataset: Data, Benchmarks and Analysis.” *ArXiv:2011.12954 [Cs]*.

Karkare, Prateek. 2019. “Handling Missing Values in Data.” *Medium*. Retrieved December 27, 2020 (https://medium.com/x8-the-ai-community/handling-missing-values-in-data-54e1dc77e24f).

Perez, Sarah. 2018. “Facebook Rolls out More API Restrictions and Shutdowns.” *TechCrunch*. Retrieved December 16, 2020 (https://social.techcrunch.com/2018/07/02/facebook-rolls-out-more-api-restrictions-and-shutdowns/).

Twitter. 2019. “Twitter: Monthly Active Users Worldwide.” *Statista*. Retrieved December 13, 2020 (https://www-statista-com.ezproxy.fhstp.ac.at:2443/statistics/282087/number-of-monthly-active-twitter-users/).

List of Figures

List of Tables

List of Listings

Appendix

A. Full table showing all researched APIs and services

As the table is too large to be appended in its entirety, it is split up into three pieces.

Part 1:

Name	Type	Pricing	Authentication	URL
Twitter	Service	Free but with limits	Yes, API keys, complicated developer account	https://twitter.com
Instagram	Service	Free	Yes, user account without 2FA	https://instagram.com
sensor.community	Service	Free	No	https://sensor.community/en/
Wheelmap.org / accessibility.cloud	Service	Free	Yes, API keys	https://wheelmap.org https://www.accessibility.cloud
OpenStreetMap	Service	Free	No	https://www.openstreetmap.org
Mapillayer	Service	Free	Yes, API keys	https://www.mapillary.com/
Foursquare	Service	Free	Yes, API keys	https://developer.foursquare.com
Weather stack	Service	Free but with limits	Yes, API keys	https://weatherstack.com
Google Street View	Service	Costs something	Yes, API keys	https://google.com/maps
Google Satelites Image	Service	Costs something	Yes, API keys	https://google.com/maps
Bing Maps Satelites Image	Service	Free	Yes, API keys	https://bing.com/maps
Mapbox	Service	Free, costs maybe something	Yes, API keys	https://mapbox.com
Here Maps	Service	Free but with limits	Yes, API keys	https://developer.here.com
Flickr	Service	Free	Yes, API keys and user account	https://www.flickr.com
Copernicus	Service	Free	Yes, user account	https://scihub.copernicus.eu/
Google Elevation	Service	Costs something	Yes, API keys	https://google.com/maps
Elevation Geoland	Dataset	Free	No	http://geoland.at
Ambient Noise	Service	Free	No	https://www.laerminfo.at/laermkarten/methoden/inspire.html
CORINE Land Cover	Service	Free	No, only for Download	
LiSA	Dataset	Free	No	https://www.landinformationsystem.at/
Reported Noise Data	Dataset	Free	No	

Part 2:

Name	Category	Description	In Use	Region
Twitter	Social Media	Realtime Status Updates	Yes	Worldwide
Instagram	Social Media	User Generated Photos	Yes	Worldwide
sensor.community	Environment	PM10, Luftqualitäts-Index (US), Temperatur, rel. Luftfeuchte, Luftdruck, Lautstärke	No	DACH
Wheelmap.org / accessibility.cloud	Information	Accessibility of places	No	Worldwide
OpenStreetMap	Environment, Information	POIs, businesses, landuse	Yes	Worldwide
Mapillayer	Images	User Capturesd Images, Pano and 2D	Yes	Worldwide
Foursquare	Information	POIs, businesses	No	Worldwide
Weather stack	Environment	Weather, free only real time, costs min. 10$ for historical data	Yes	Worldwide
Google Street View	Images	200€ per month are free credits, doesn't return 360° images, can be generated from cubemap, max resolution is 640x640	Yes	Worldwide
Google Satelites Image	Images	200€ per month are free credits, max resolution is 1280x1280	Yes	Worldwide
Bing Maps Satelites Image	Images	Satelite Images, high resolution, low quality	Yes	Worldwide
Mapbox	Images	Satelite Images	No	Worldwide
Here Maps	Images	Satelite Images, Terrain Images	No	Worldwide
Flickr	Social Media	User Generated Photos	Yes	Worldwide
Copernicus	Environment, Images	Sentinel-1, Sentinel-2, Sentinel-3 and Sentinel-5P user products	No	Worldwide
Google Elevation	Environment	Elevation Data worldwide	No	Worldwide
Elevation Geoland	Environment	Elevation GeoTIFF for Austria in 10m resolution	Yes	Austria
Ambient Noise	Environment	Noise Data for streets, railway, planes, industry	Yes	Austria
CORINE Land Cover	Environment	Land use (Land cover) as WMS service, GeoTIFF available as download	No	Europe
LiSA	Environment	Land cover, Landuse only for small parts, not usable	No	Austria
Reported Noise Data	Environment	Excel containing data for some cities, regions, not usable	No	Europe

Part 3:

Name	Docs
Twitter	https://developer.twitter.com/en/docs/twitter-api/tweets/search/quick-start https://twitter.com/search?q=geocode%3A48.200843%2C16.352585%2C.05km&f=live
Instagram	https://github.com/ping/instagram_private_api https://instagram-private-api.readthedocs.io/en/latest/api.html#instagram_private_api.Client.location_search https://instagram-private-api.readthedocs.io/en/latest/api.html#instagram_private_api.Client.feed_location
sensor.community	https://github.com/opendata-stuttgart/meta/wiki/EN-APIs
Wheelmap.org / accessibility.cloud	https://github.com/sozialhelden/accessibility-cloud/blob/master/app/docs/json-api.md
OpenStreetMap	https://wiki.openstreetmap.org/wiki/Overpass_API Landuse: https://wiki.openstreetmap.org/wiki/Landuse
Mapillayer	https://www.mapillary.com/developer/api-documentation/#images
Foursquare	https://developer.foursquare.com/docs/places-api/ https://developer.foursquare.com/docs/api-reference/venues/search/
Weather stack	https://weatherstack.com/documentation
Google Street View	https://developers.google.com/maps/documentation/streetview/overview https://colab.research.google.com/drive/1WcpbHxeqKdzeHo5t9Si4GRUgRYrSFPFT
Google Satelites Image	https://developers.google.com/maps/documentation/maps-static/start
Bing Maps Satelites Image	https://docs.microsoft.com/en-us/bingmaps/rest-services/imagery/get-a-static-map https://www.bingmapsportal.com/
Mapbox	https://docs.mapbox.com/api/maps/#static-images https://docs.mapbox.com/playground/static/
Here Maps	https://developer.here.com/documentation/map-image/dev_guide/topics/resource-map.html
Flickr	https://www.flickr.com/services/api/flickr.photos.search.htm https://colab.research.google.com/drive/1rStpxZtula8vNLe2gTlbaqhL9JNQOfEE
Copernicus	https://scihub.copernicus.eu/userguide/WebHome https://scihub.copernicus.eu/twiki/do/view/SciHubUserGuide/OpenSearchAPI#Open_Search_Queries_examples https://scihub.copernicus.eu/dhus/search?q=footprint:"Intersects(48.200710,16.352582)"
Google Elevation	https://developers.google.com/maps/documentation/elevation/overview
Elevation Geoland	https://www.data.gv.at/katalog/dataset/d88a1246-9684-480b-a480-ff63286b35b7
Ambient Noise	https://geometadatensuche.inspire.gv.at/metadatensuche/srv/ger/catalog.search#/metadata/36de2b6e-5339-4377-b5d-bf8b1c5f953b https://inspire.lfrz.gv.at/000804/ows?service=wms&request=GetCapabilities
CORINE Land Cover	https://land.copernicus.eu/pan-european/corine-land-cover/clc2018?tab=mapview https://image.discomap.eea.europa.eu/arcgis/rest/services/Corine/CLC2018_WM/MapServer
LiSA	https://www.landinformationsystem.at/#/cadaster-env/overview
Reported Noise Data	https://www.eea.europa.eu/data-and-maps/data/data-on-noise-exposure-7/