Chapter 3: System architecture

Draft version

This chapter describes the methodology of DBAFS. It builds on the theory discussed in Chapter 2, and is structured as follows. In the first section, a general overview of the complete forecasting system is given. Section two presents the software that underlies DBAFS, while in the third section, the computations on the database server are discussed. The last three sections cover the detailed methodologies of all the distinct components of the system architecture seperately.

3.1 Overall design

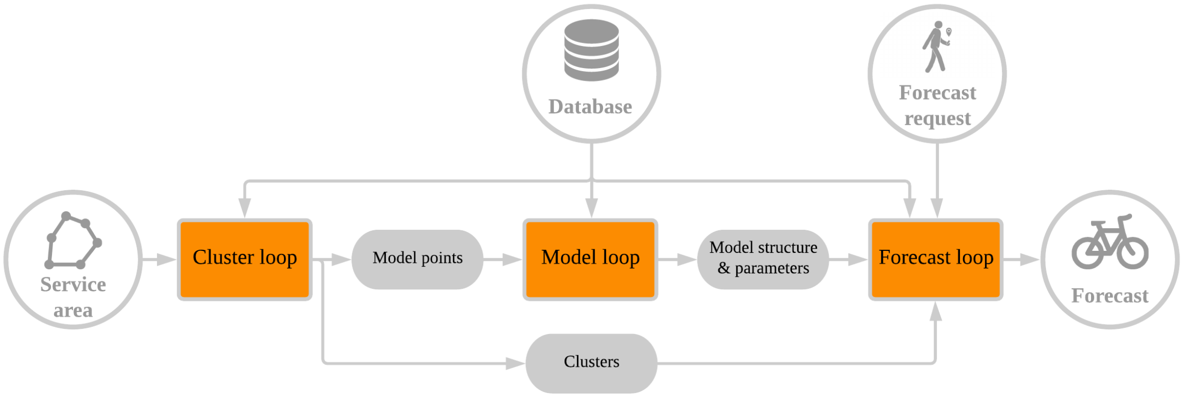
The goal of DBAFS is to forecast the distance to the nearest available bike for a given location and a given timestamp in the future. It is meant to be used by both the operators and users of a dockless bike sharing system, which from now on are refered to as *users* of DBAFS. A forecast is made every time a user requests one. In intensively used bike sharing sytems, this can mean that several hundreds of forecasts are required every day, all based on different historical datasets. All these datasets usually consist of a time series with a high temporal resolution. Although the data may be complex, it would be inconvenient for the users if forecasts take a lot of time or need manual interventions. Taking into consideration the above-mentioned challenges, DBAFS should be a *fast* and *automated* process that still produces as accurate forecasts as possible.

The most time consuming part of the system is the selection of an appropriate model and the estimation of its parameters. If this had to be done at every forecast request seperately, forecasts would take too much time. Therefore, in DBAFS, forecasting models are build only once in a while at a limited number of locations. Each individual forecast will inherit the structure and parameters of one of those pre-build models, rather than building a completely new model on its own.

The approach of building models only at a limited number of locations, involves the selection of those locations. In DBAFS, this is done by dividing the service area of the dockless bike sharing system into spatially contiguous clusters, where each cluster contains the areas that show similar weekly patterns in the historical data. Then, each cluster is represented by a single *model point*, which is a location where a model is build. An individual forecast takes the model structure and paramaters of the model point that is in the same cluster as the location of the forecast.

To summarize, DBAFS consists of three main processing loops. In the cluster loop, the service area is divided into clusters based on weekly patterns in the historical data. In the model loop, forecasting models are build on the historical data of one location per cluster. In the forecast loop, forecasts are made by using the model structure and parameters of the model that was build in the same cluster as the location of the forecast. The forecast loop runs every time a user makes a forecast request. The model loop only runs every n_m weeks. n_m should be chosen such that new models are build when the patterns in the historical data have changed considerably. Based on real-world data, which are presented later in this thesis, it is recommended to update the models every month. That is, $n_m = 4$. The cluster loop runs every n_c weeks. n_c should be chosen such that new clusters are defined when the spatial distribution of the weekly patterns in the historical data has changed considerably. The real-world data used in this thesis were not long enough to define a recommended value for n_c, but in most cases, it will be much larger than n_m. The cluster, model and forecast loops are all completely automated and do not require any manual interventions.

The overall design of DBAFS is summarized graphically in Figure 3. The detailed methodologies of the three loops are discussed in section 3.4, 3.5 and 3.6, respectively.



3.2 Software

The underlying code of DBAFS is written in the R programming languague (R Core Team 2013). However, Structured Query Language (SQL) statements are nested within the R code to retrieve data from a PostgreSQL database (PostgreSQL 2014), and run some of the heavier data pre-processing computations on the database server. These computations are discussed in the next section.

On top of functions that are included in R by default, DBAFS makes use of several extentions, as listed below.

- The `ClustGeo` package, for spatially constrained clustering (Chavent et al. 2018).
- The `forecast` package, for building forecasting models, decomposing time series, and forecasting time series (R. J. Hyndman and Khandakar 2008).
- The `lubridate` package, for processing dates and timestamps (Grolemund and Wickham 2011).
- The `RPostgreSQL` package, for connecting to a PostgreSQL database and running SQL code on the database server (Conway et al. 2017).
- The `sf` package, for processing spatial data (Pebesma 2018).
- The `tsibble` package, for pre-processing time series datasets (Wang, Cook, and Hyndman 2018).

3.3 Database

In a dockless bike sharing system, each bike is equipped with a Global Positioning System (GPS). Every i_d minutes, the geographical locations of all bikes are saved into a database, together with the corresponding timestamp. The locations of the bikes that are not in use at the current time, and thus available, are usually visible to the users of the system in a mobile application, and stored seperately from the data regarding bikes that are in use.

The geographical location of a bike is spatial data, and should be stored as such. An advanced and open source database management system for spatial data is PostgreSQL in combination with the PostGIS extension. DBAFS requires the data to be stored in such a database. Each feature represents the location of an available bike at a certain timestamp and should at least have the following fields.

- A timestamp of data type `timestamp with time zone`.
- A geographical location of data type `geometry(Point)`.
- A unique ID of the bike to which the feature belongs.

Data are pre-processed on the database server, and only the data that are needed, are loaded into memory. In DBAFS, this pre-processing step involves two different procedures. The first one leads to data that contain information about the distance to the nearest bike for several timestamps in the past, and is discussed in the next sub-section, while the latter produces a dataset with all the bicycle pick-ups in the database, and is discussed in section 3.3.2.

3.3.1 Distance data

For a given location, the distance from that location to the nearest available bike is calculated for each timestamp $t \in T$, where T is a regulary spaced time interval containing timestamps that are present in the database. The temporal resolution of T equals i_s minutes, where $i_s \geq i_d$. The nearest available bike is found by a nearest neighbor searching process that uses spatial indices on the geometries. In practice, this means that it is not needed to first compute the distances to all available bikes, which would slow down the process vastly.

The calculated distances are great-circle distances assuming a spherical earth a radius equal to the mean radius of the WGS84 ellipsoid, as showed in Equation x.

$$L_{AB} = \frac{(2a+b)}{3} \times \frac{\pi}{180} \times arccos(sin\phi_A sin\phi_B + cos\phi_A cos\phi_B cos\Delta\lambda)$$

Where L_{AB} is the great-circle distance between point A and point B in meters, ϕ_A and ϕ_B are the latitudes of respectively point A and B in degrees on the WGS84 ellipsoid, and $\Delta\lambda$ is the difference in longitude between the two points, i.e. $\lambda_B - \lambda_A$, in degrees on the WGS84 ellipsoid. Furthermore, a is the equatorial radius of the WGS84 ellipsoid in meters, which is defined to be 6378137, and b is the polar radius of the WGS84 ellipsoid in meters, which is defined to be $6378137 \times (1 - 298.257223563^{-1}) = 6356752.3142$ (Iliffe and Lott 2008).

The sphere is chosen since calculating distances on the ellipsoid itself slows down computations, and, on the geographical scale of a dockless bike sharing system, has an accuracy gain that can be neglected. Working with the shortest distance over the street network might in most cases be more approriate, but at the same time involves much more complex computations, especially when either the given location or the locations of the bikes are not exactly on the network lines.

The output of this pre-processing operation is a time series with T features and a temporal resolution of i_s, belonging to one single location in the service area of the dockless bike sharing system. Each feature contains a timestamp and the great-circle distance from the given location to the nearest available bike in meters. Such data are refered to in this thesis as *distance data*.

3.3.2 Usage data

A pick-up is the moment that a user of the dockless bike sharing system unlocks a bike to make a trip. For the historical database containing the locations of the available bikes, this means that the bike that is picked-up will be present in the data at the last timestamp before the pick-up, but missing at the first timestamp after the pick-up. In DBAFS, this is used to retrieve all the pick-ups from the database. Historical data with the highest possible temporal resolution, i.e. i_d minutes, are queried for one single bike ID. Then, all timestamps that are missing, are added to the data, but without an available location. If feature j has an available location, but feature $j+1$ has not, j is considered a pick-up. This procedure is repeated for all individual bikes.

The output of this pre-processing operation is a data frame with all the features in the database that are considered pick-ups. Each feature has at least a timestamp, a geographical location and a bike ID. The number of pick-ups in an area represents the usage intensity of the bike sharing system. Such data are therefore refered to in this thesis as *usage data*.

Obviously, the procedure described in this sub-section has some deficiencies. The start of a GPS failure is falsely considered to be a pick-up, just as the removal of a bike by the system operator, for redistribution or maintenance purposes. Specific information about redistribution patterns can be added, but will in many cases be inavailable, and even if available, those patterns may be too irregular to implement adequately in the workflow. However, in DBAFS, the usage data are only used to find appropriate locations for the model points, and not to analyze usage patterns into detail. Therefore, fully accurate data are not indispensable for this purpose, and the current procedure is sufficient.

3.4 Cluster loop

3.5 Model loop

3.6 Forecast loop

Chavent, Marie, Vanessa Kuentz-Simonet, Amaury Labenne, and Jérôme Saracco. 2018. "ClustGeo: an R package for hierarchical clustering with spatial constraints." *Computational Statistics* 33 (4): 1799–1822. doi:10.1007/s00180-018-0791-1.

Conway, Joe, Dirk Eddelbuettel, Tomoaki Nishiyama, Sameer Kumar Prayaga, and Neil Tiffin. 2017. *RPostgreSQL: R Interface to the 'PostgreSQL' Database System.* https://cran.r-project.org/package=RPostgreSQL.

Grolemund, Garrett, and Hadley Wickham. 2011. "Dates and Times Made Easy with lubridate." *Journal of Statistical Software* 40 (3): 1–25. http://www.jstatsoft.org/v40/i03/.

Hyndman, Rob J, and Yeasmin Khandakar. 2008. "Automatic time series forecasting: the forecast package for R." *Journal of Statistical Software* 26 (3): 1–22. http://www.jstatsoft.org/article/view/v027i03.

Iliffe, J, and R Lott. 2008. *Datums and Map Projections: For Remote Sensing, GIS and Surveying.* Whittles Pub.

Pebesma, Edzer. 2018. "Simple Features for R: Standardized Support for Spatial Vector Data." *The R Journal.* https://journal.r-project.org/archive/2018/RJ-2018-009/index.html.

PostgreSQL. 2014. "PostgreSQL: The world's most advanced open source database." http://www.postgresql.org/.

R Core Team. 2013. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. http://www.r-project.org/.

Wang, Earo, Di Cook, and Rob Hyndman. 2018. *tsibble: Tidy Temporal Data Frames and Tools.* https://pkg.earo.me/tsibble.