Predicting Parking Space Availability in Paderborn

Jan Lippert <ljan@mail.upb.de>

***Abstract*—The Parking Prediction system Paderborn is available at http://pppb.herokuapp.com/. It was written in Play! Scala and uses the haifengl.github.io/smile/ library. the system regularly predicts the number of available parking spaces for the "Libori-Galerie" 15 minutes in advance. The predictions are still not perfect but turned out to be quite good.**

I. Introduction

A. What question does your work answer

In urban areas, space and as such parking space is limited. Car owners often have to drive around to available parking spots. This leads to an unnecessary waste of time and additional air pollution.

According to multiple studies, the availability of parking data reduces the search time for a parking spot [1][2]. However, most systems only provide the current number free parking spots. In rush hours, this information can quickly get outdated and lead to driving around to find available parking spaces.

In Paderborn the situation is similar: live-data is available but no prediction. The goal of this project is exactly that. Because of the limitation outlined in V, data-crawling and future predictions will focus on the covered car park "Libori-Galerie". This car park is also the most interesting as it is directly connected to a local shopping center.

B. Related Work

[3] is a similar project which used the collected data to predict parking space availability. This project focused on the car park "Centrum Galerie". [3] used the following features: week of year, day of week, time of day, Sunday openings and the number of workdays until the next public holiday. Sunday openings are used as in Germany, shops are normally closed on Sundays. Sunday openings therefore influence how many people go to the shopping center and therefore the availability of parking space. In [3], different algorithms are compared and the predictions are evaluated.

[4] propose parking guiding and information system which also includes the prediction of available parking space. The proposed algorithm uses a probabilistic model based on historical data. Among others, the used features also include time of day and day of week. At last, [4] estimates the mean-error on predictions. They mention that predictions for 10 minutes in the future lead to a 1.2% error on average while predictions for 40 minutes in the future lead to a 2.8% error on average.

[5] compared three different feature sets and three different machine learning algorithms – regression tree, neural network, and support vector regression – with respect to their performance. Based on data from the cities of Melbourne and San Francisco they conclude that the regression tree provides the best predictions when used with a feature set containing day of week and time of day.

II. Methodology choice

Since this work is inspired by [3], a RegressionTree model was chosen as machine learning method. To keep the complexity of the application in check, the model was kept simple: the data consisted only of timestamped availability data.

It turned out that the initial choice of using a RegressionTree performed very poorly on the first crawled data sets. Therefore experiments to compare different models were conducted. Over the course of the project, it was decided to choose the model anew on every prediction by picking the best model out of a set of different ones. Finally, it turned out that GradientTreeBoost is the best performing model.

A. Data Sources

The main data source is the actual parking data which is available online. In addition there are other possible features that may influence parking space availability. Since taking all features into account is not possible, this project will solely focus on the parking data in relation to time.

a) Parking Usage Live Data: The homepage of the ASP Paderborn[1] lists the city managed parking areas and the number of currently free parking spots. The data is available on https://www.paderborn.de/microsite/asp/parken_in_der_city/freie_Parkplaetze_neu.php. A more minimal website is available at https://www4.paderborn.de/ParkInfoASP/default.aspx.

The website displays 4 attributes: name, type, capacity, and available spots[2]. After crawling, the parking information will be annotated with the crawling time. The latter will be split into multiple attributes which will be used in the prediction: hour of day h, minute of hour m, day of week d, day of month d_m, week of month w_m, week of year w_y, and a[3].

The number of available parking spaces is be called y. With the features defined before, the training data will be a set of tuples of the form

$$(h, m, d, d_m, w_m, w_y, a, y).$$

b) Other data-sources: It was also considered to enrich the data set by adding event and holiday data as both could influence the parking situation near the city center. In some cases, these events take place on one of the parking areas or directly besides them. However, as most of the big events do not happen while the course "Practical Project in Machine Learning" takes place, they will not considered further.

[1]City-Managed service for waste management, city cleaning, and parking.

[2]respectively: Parksttte, –, Anzahl, Frei

[3]the year is included as a way to keep samples unique in case the project runs more than 1 year.

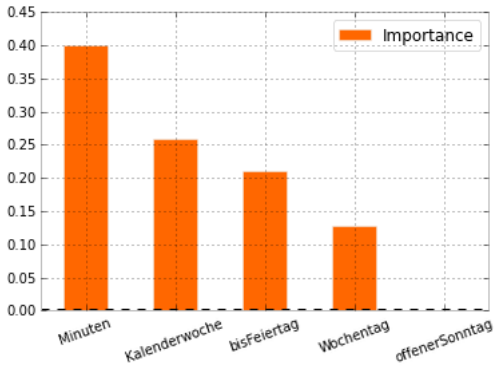


Fig. 1. Importance of Features (ParkenDD)

Public holidays like Christmas and Easter may also influence the availability of parking space. [3] modeled this in the feature "bisFeiertag", i.e. working days until public holiday. In figure 1, the relevance of the different features used by [3] is displayed[4].

The feature was relatively important in the evaluation done by [3]. Nonetheless, it will not be included as the next public holiday is Christi Himmelfahrt on 25th May 2017. Therefore there would be no possibility of verifying the importance of this feature in time.

Weather data was also considered as an additional source of information because, e.g., more people may decide to take the car when it's raining. Although such data is available online in machine-readable formats, it will not be considered to keep the model simple.

B. Hypothesis & Training Method

It is assumed, that the available parking space is dependent on time, i.e. there exists a function f such that

$$f(h, m, d, d_m, w_m, w_y, a) = y.$$

As mentioned in previous sections, the idea to predict available parking space was inspired by other projects. [3] was using a RegressionTree model with great success, so I picked this model.

To keep the scope of this project small, multiple data sources will be ignored. For this project, it is assumed that the number of available parking spots solely depends on time. As described in II-A, this obviously is a very simplified model. However, as shown by [3], such a simple model may be enough.

C. Comparison between methodologies, if applicable

The smile library provides different learning models for Linear Regression. After a few experiments, the predictions of three models – GradientTreeBoost, RandomForests, and RegressionTrees – were "not completely off". Since I do not have any previous experience in Machine Learning, the prediction module was implemented to train evaluate multiple models with different parameterizations on the crawled data and pick the one with the minimum mean average error to make a prediction.

Currently, there are 8567 predictions in total[5]. In the course of the project GradientBoostTrees turned out to be the dominating prediction model with 7755 predictions. Only 99 of the predictions made by the system were derived from RandomForests and 713 were derived from using a RegressionTree. In the latest 48 hours of predictions, only GradientBoostTrees were chosen as the best model for prediction.

[4]respectively: minutes, week of year, working days until public holiday, day of week, Sunday openings

TABLE I
EVALUATION PER DAY OF YEAR

$date$	n	MAE	σ
01.03.	103	26.14	36.35
02.03.	185	23.64	29.54
03.03.	286	21.24	24.80
04.03.	521	39.06	36.61
05.03.	1091	67.32	47.67
06.03.	1082	51.28	45.73
07.03.	925	204.89	127.97
08.03.	969	186.90	172.56
09.03.	1050	20.70	28.86
10.03.	1085	25.24	27.12
11.03.	586	30.31	40.46

III. VERIFICATION

The functionality of the system will be verified by doing regular predictions and comparing these to the actual parking situation later. Similar to the crawling implementation, the web application will generate a new prediction "15 minutes into the future". These predictions will be saved in a database.

To measure the error, we will use the mean absolute error

$$MAE := mean(|y_i - y_i|),$$

where y_i denotes the predicted value and y_i denotes the actual, crawled value. Table I lists the evaluation results fetched on 11.03.2017 around 6pm.

The evaluations show that the predictions got worse on the 4th and are especially bad on the 7th and 8th March. The cause for this was some experimentation with the training method and the model.

On the 4th, the training method was adapted. Instead of generating 10 GradientBoostTrees with shrinkage values from 0.1 to 1 (step width 0.1), only 4 GradientBoostTrees with fixed shrinkage values of (0.001, 0.01, 0.1, and 1 were generated. According to the documentation of GradientBoost, the shrinkage parameter controls the learning rate; it was assumed that picking a set of four very different values would sufficient for good predictions.

On the 7th, the model attributes "week of month" and "week of year" were removed. All Learning models compute the so-called importances of the attributes used for learning. According to the importances determined in previous predictions, the aforementioned attributes were not important. Also, from this day on the attributes were normalized when training the model. This was not necessary but I wanted to try it out anyway.

[5]Checked on 11.03.2017 at 7:30pm.

Fig. 2. Crawled parking space availability

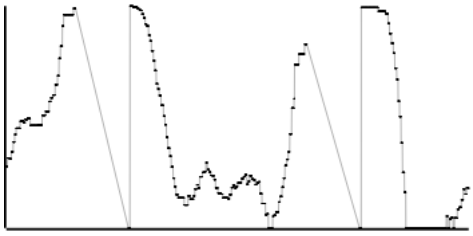
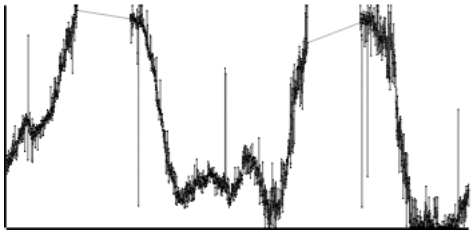


Fig. 3. Predicted parking space availability



As one can see in Table I both changes did not improve the model but instead made predictions worse. Therefore the changes were rolled back on the 9th. After this was done, the prediction accuracy increased again. Figures 2 and 3 display the actual availability respectively the predicted availability of parking spaces over the last 48 hours. As one can see, the tendency of the predictions is quite good. However, the predictions tend to overshoot and "wiggle around". When presenting the intermediate results in class, it was suggested to dampen the predictions.

IV. Run-time technology

It was chosen to develop the web application in Scala using the Play! framework, mainly because of familiarity with the language and its usage in machine learning applications. It is also used in

A. Architecture

One important part in the choice of technology was the separation of the different concerns in the applications. There are multiple parts that play together: crawling, preprocessing, training, prediction, and finally evaluation.

Crawling has to be done on a regular basis. Therefore the chosen framework needs to support regular execution of jobs. When new entries are crawled, they are cleaned via the preprocessing module and inserted in the database. Since it may be necessary to update the model over time, the preprocessing module will also have a regular job that cleans existing entries by updating them to the most recent model.

B. Frameworks

To implement this project, Ruby on Rails, Python and Scala were considered. I used Ruby on Rails in previous projects and development is quite fast with this framework. However, I could only find a few libraries that deal with machine learning [6] [7]. Another choice of language was python. Python has quite a lot pf machine learning libraries and is also used in academics. However, I do not have much previous experience with python.

My final choice was the https://playframework.com with Scala. I did use this framework in previous projects and therefore was familiar with setting up background jobs and how to enable web access. Heroku also supports easy deployment for Play! applications. One important factor of this choice was the type-safety of the Scala language and it's usage in machine learning.

I chose to use http://haifengl.github.io/smile/index.html, the "Statistical Machine Intelligence and Learning Engine". It was easy to use and documentation is quite extensive. In some cases, the API documentation even references the scientific papers the learning algorithm is based on.

V. Encountered Problems

A. Malfunction of the Parking Guidance System

One week after the first prototype of the crawler was online, https://www.paderborn.de/microsite/asp/parken_in_der_city/freie_Parkplaetze_neu.php was non-functional. The number of available parking spaces for the "Libori-Galerie" was always set to 0.

At the same time, many of the other parking areas were switched to "Nicht im Parkleitsystem" (not part of the parking guidance system). Both of these issues were caused by a malfunction of the parking guidance system. Crawling was resumed normally after the parking guidance system was fixed by the provider.

On another note, string values were not expected for the Libori-Galerie. This caused the crawler to crash on every crawl; the crawler was then adapted to be ore resilient to unexpected values: all non-integer values for free spaces will directly be dropped.

B. Heroku Free Limitations

After collecting data for some time, I did some correlation analysis on the data. I noticed that all entries had "$w_m = 1$" despite the app being online for more than a week. Further investigation showed that the data was only available for 3 different days.

The reason for this failure was the uninformed use of the Heroku Free package. Free dynos will sleep after 30 minutes minutes of inactivity. After this was noticed, multiple services and workarounds were investigated.

However, most of the workarounds were from before 2015. In 2015, Heroku added the requirement that free dynos need to sleep 6 hours a day. Unfortunately the most promising service – http://kaffeine.herokuapp.com/ – is not functional anymore.

To keep the system running, a bash script was created which keeps the dyno awake by sending regular HEAD requests. Since the dyno is still required to sleep 6 hours a day, a sleeping time had be chosen. Luckily, the "Libori-Galerie" is closed from 2am to 8am and the dyno will rest in this period.

Also, there were multiple problems in accessing the database The free tier only allows 20 concurrent connections to the Postgresql database. These connections were exhausted 5 minutes after the start of the application. This could be

solved by configuring the application to restricting the number of database connections to 10 – when the application was configured to use 20 connections, problems persisted.

The next limit is in the size of the database. The free tier only allows 10000 rows. Insert actions will be disabled after the database has had more than 10000 for 7 days. This challenge will be circumvented by removing old data in regular intervals.

VI. Reflection

The most important lesson I learned was: do more regular checks of a deployed system. Many of the problems were discovered too late because "it worked on my local machine". All of these could have been avoided by checking the web app more regularly or setting up some service to check the system.

Another important lesson was that doing machine learning without previous experience is hard. This was my first machine learning course; learning the methods and implementing at the same time was very challenging. However, the gained experience was worth it.

References

[1] Y. Asakura and M. Kashiwadani, "Effects of parking availability information on system performance: a simulation model approach," in *Vehicle Navigation and Information Systems Conference, 1994. Proceedings., 1994*, pp. 251–254, Aug 1994.

[2] F. Caicedo, "Real-time parking information management to reduce search time, vehicle displacement and emissions," *Transportation Research Part D: Transport and Environment*, vol. 15, no. 4, pp. 228 – 234, 2010.

[3] P. Balzer, "Vorhersage der parkhausbelegung mit offenen daten," Mar. 2015. Available: http://mechlab-engineering.de/2015/03/vorhersage-der-parkhausbelegung-mit-offenen-daten/ [Accessed: 24.01.2017].

[4] T. Rajabioun, B. Foster, and P. Ioannou, "Intelligent parking assist," in *21st Mediterranean Conference on Control and Automation*, pp. 1156–1161, June 2013.

[5] Y. Zheng, S. Rajasegarar, and C. Leckie, "Parking availability prediction for sensor-enabled car parks in smart cities," in *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pp. 1–6, April 2015.

[6] J. Donaldson, "Machine learning on rails with ruby!," 2012. Available: https://blog.bigml.com/2012/07/06/machine-learning-on-rails-with-ruby/ [Accessed: 22.02.2017].

[7] L. Masini, "Machine learning made simple with ruby," Aug. 2015. Available: https://www.leanpanda.com/blog/2015/08/24/machine-learning-automatic-classification/ [Accessed: 22.02.2017].