Occupy NTK

Václav Maixner
ČVUT–FIT
maixnvac@fit.cvut.cz

May 17, 2020

Strukturu sekcí reportu si pochopitelně můžete upravit, aby vám co nejlépe vyhovovala pro popis vaší práce.

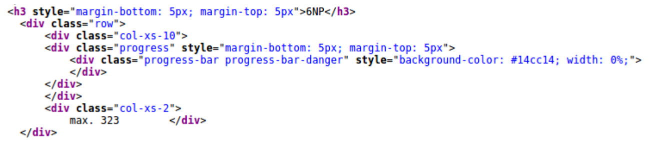
1 Introduction

Národní technická knihovna (NTK) offers a large and free space for students from all universities to study in. All of its great properties are, however, often outweighed by it's major weakness - too many students. It takes 27 min to get from Strahov dormitory to NTK, 15(5) min to aimlessly wonder the 4 floors for a free sitting space, only to find none and return to Strahov.

This waste of time can be prevented by having the information about current occupation of NTK. This is provided on their website [1], but in a very not-friendly way and with no history provided. The aim of this Flask application is to show more detailed information in a user-friendly way and provide the users with history of NTK's occupation, allowing for better decisions before travelling to NTK.

2 Getting the data

As mentioned above, the data is available at NTK's website, shown as occupational percentage of individual floors along with the maximum number of seats shown next to it. The graphs do not show the actual percentage value, leaving the user guessing as to what the actual percentage is. As can be seen in figure 1, the percentage value is passed to the Bootstrap progress bar element.



```
<h3 style="margin-bottom: 5px; margin-top: 5px">6NP</h3>
  <div class="row">
      <div class="col-xs-10">
      <div class="progress" style="margin-bottom: 5px; margin-top: 5px">
          <div class="progress-bar progress-bar-danger" style="background-color: #14cc14; width: 0%;">
          </div>
      </div>
      </div>
      <div class="col-xs-2">
          max. 323        </div>
  </div>
```

Figure 1: Sample of NTK html code.

The percentages are scraped periodically by, scheduled by `APScheduler`. `Requests` are used to put the GET request to NTK's website, which is then parsed by `Beautiful Soup`. The progress bar elements have unique class names, making it easy to parse them. The percentage values are then retrieved using regular expressions.

3 Program structure

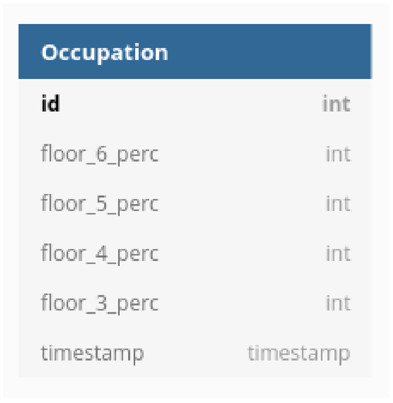
The scraper is called from a Flask application, the structure of which is inspired by Miguel Grinberg as described in [1]. Flask is one of the two most popular web frameworks along with Django and is often classified as a micro-framework. It's no-batteries-included approach gives the programmer a lot of freedom and it's lightness is advantageous for rapid deployments of simple apps. It uses the MVC design pattern.

The flask app stores the scraped data in an SQLite database. Despite SQLite not being a grown-up database, it's simple use and lightweight nature make it an ideal first choice for the simple operations needed for this project. The database is interacted with via SQLalchemy, an ORM. Since SQLalchemy doesn't support changes to Models (equal to table alteration), Alembic has been used to manage different versions and allow for transitions during the development period. Since the app is not yet in production, no user authentication has been employed yet.

There is only one table in the database, with percentages of occupation of the 4 floors, as in figure 2. The overall occupation, and the conversion to number of visitors is provided by helper get functions. The table then has unique `id` column and `timestamp`, with the time of creation. The data is scraped every 5 min, long enough not to make too many requests and short enough to provide up-to-date information.

Flask, unlike Django, doesn't come with Admin interface included, so library Flask-Admin has been utilized. It provides overview of the data in database, making it easier to review the data during development process when compared to terminal. It can be accessed at `http://localhost:PORT/admin`.

[1] At `https://www.techlib.cz/cs/83028-mista-ke-studiu`



Occupation	
id	int
floor_6_perc	int
floor_5_perc	int
floor_4_perc	int
floor_3_perc	int
timestamp	timestamp

Figure 2: SQL table blueprint.

4 Plotting

The target of this app is to provide two services to the user - current occupation and information about previous occupation. The history is plotted on graphs, with the target being dynamic graphing to keep them graphs up-to-date. The most prominent Python plotting library is Matplotlib. The way to use it would be to query the data from database, plot the graphs and print the png file to server. When the user visits the website, the image is served.

This has, however, the disadvantage of creating static only figures. Bokeh is a Python library that leverages Python's ease of use and generates Javascript code. It allows for dynamic plotting, making it more viable option for possible future growth of the application's features. Bokeh also seamlessly integrates with Matplotlib and Seaborn.

5 What's it good for?

The app can provide more information and plot it for the user to see. It also deals with a certain bias in NTK's data - when the capacity is reaching above ≈60 %, the 3rd floor starts to register above 100 % occupation. With increasing number of people in NTK, this situation gets only worse. The graphs provided by their website however stop at 100 %, often showing the 3rd floor as full and the other floors as relatively empty.

This is however not true, as the 150 % of 3rd floor visitors are spread out over the other floors. This is probably due to a measuring error - visitors are counted by step counters at the beginning of each stairs leading to the floor above. This is further worsened by the usage of lift. This app provides the information on overall occupancy, which yields much more accurate prediction.

What is also of immense help is the history. Students from across multiple universities visit NTK and their examination periods differ greatly. The graphs help to answer 3 basic questions:

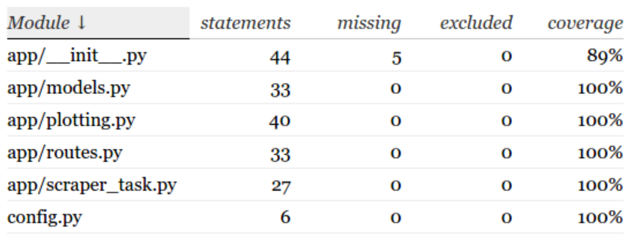
1. How bad is it today?
2. How bad was it yesterday after this time?
3. How bad was the past week?

The last graph offers more insight into what the current situation is, so that one doesn't have to look up the examination period of Faculty of Law every year, twice. As the data accumulates, seasonal forecasting using ARIMA could be used to roughly predict the occupancy during the upcoming exams, allowing the students who can be productive only in NTK to plan their terms around the times the NTK is choke full of people.

This has however been hindered by the current COVID-19 crisis. The app currently runs on dummy data generated for the needs of testing. Any forecasting has to be done on actual data we want to forecast, thus this part has been postponed until NTK reopens and enough data is acquired.

6 Tests

The app has unit tests testing the Model and the scraper. For running the tests, the Pytest library was used. To review the coverage of these test, Coverage library was employed, the result of which can be seen in 3, with 89 % test coverage. This result is rather peculiar, as plotting itself has not been tested. The `__init__` has one uncovered function, which is used to drop all data from database and refill it with predefined test data. As this is in itself a method used for testing purposes, not for production, it has not been tested.



Module ↓	*statements*	*missing*	*excluded*	*coverage*
app/__init__.py	44	5	0	89%
app/models.py	33	0	0	100%
app/plotting.py	40	0	0	100%
app/routes.py	33	0	0	100%
app/scraper_task.py	27	0	0	100%
config.py	6	0	0	100%

Figure 3: Unit tests coverage of the app.

The tests include the `test_basic`, which is more of an integration test, used to check that the app is running and Pytest can find the test files. In the future, the project should be refactored to separate the frontend from the backend, making testing much easier. Part of this process should be introduction of integration tests, as they help test that the user receives the final product. They also mitigate the problem of the code's author forgetting to test certain scenarios.

7 Review

The app works and despite it's simplicity can already provide enough information for the user to make an educated guess, rather than a guess. Once NTK reopens and up-to-date data is scraped, predictions can be introduced into it.

References

[1] M. Grinberg. *Flask Web Development: Developing Web Applications with Python.* O'Reilly Media, 2018.