Data. ACHTUNG NICHT FERTIG!

Thure Nebendahl

May 22, 2020

This chapter discusses in more detail the discrete data stream used as a basis for the investigation of the bounded history encoding from [1]. It explains where this data comes from, how it is retrieved and how it can be processed. Finally, it explains how the implementation of the bounded history encoding from [1] can query the data.

The publicly accessible data, which is used as a discrete data stream, comes from the online car market "AutoScout24" (AS24) [2]. It provides data on many models of various brands. The brands and models must be narrowd down so that the data remains manageable but still delivers noticeable results. It is assumed that it is irrelevant which exact brands and models are collected. LATER ON IT CAN BE SHOWN WETHER THIS IS TRUE . A COMPLETE LIST OF ALL MODELS CAN BE FOUND IN THE APPENDIX . To create a real discrete data stream, we collect the data weekly for SEVERAL weeks. This is the longest possible period for this work. TO CHECK IF THIS AFFECTS THE RESEARCH RESULT, THE RESULT FOR PARTIAL PERIODS WILL BE COMPARED WITH THE RESULTS FOR THE ENTIRE PERIOD .

The data is collected by a scraper implemented in Python, similar to the one presented in [3]. The scraper uses the modules Beautiful Soup 4 (BS4) [4], pandas [5] and SQLalchemy [6].

First, it is specified which aspects of an individual car the scraper should cover. This is necessary because most of the data collected from a website does not contain any information about the car. Again, it is assumed that it is irrelevant which exact data is collected about the car. A COMPLETE LIST OF ALL ASPECTS CAN BE FOUND IN THE APPENDIX . Second, the scraper searches AS24 for a specific brand and model and saves the link to each offer the search yields. This is performed with the BS4 "Soupstrainer" from [4]. To ensure a consistent order of the offers, they are sorted by age, starting with the newest. On AS24, the search results consist

of a maximum of 20 pages, which then themselves contain a maximum of 20 offers. However, this maximum of 400 offers per search should not have a major impact on our research results. On the one hand, it helps to keep the data manageable, on the other hand the number of brands creates a data set with over 20,000 offers per week, which is expected to be sufficiently large to deliver noticeable results. Third, once all links are saved, the scraper searches every offer individually and extracts all the previously determined aspects from the HTML, again using BS4 [4]. Fourth, the collected data is stored in a pandas "DataFrame" [5] to be loaded further into a SQLite database with SQLalchemy [6].

The limit of 400 offers per search presents some challenges. With some models, offers are added with a high frequency, so the 400 offers found since last week may all be new. To ensure the scraper searches every offer from last week again, it scans through all previously found offers before searching for new ones. At the same time, reviewing each offer provides a better reference to check which offers have been deleted. If the scraper tries to search a deleted offer, an error message is returned. This provides more certainty than just assuming an offer has been deleted if the search results no longer contain that offer. The latter could be a consequence of frequently added new offers. To simplify matters, it is assumed that every deleted offer has been sold. Although this distinction may be very important in market analysis, it is not relevant here.

THE COMPLETE SCRAPER CAN BE FOUND ON GITHUB .

The final step of storing the data in a SQLite database is required because the implementation of the bounded history encoding also operates on a SQLite database. This enables cross-language interaction between the scraper and the implementation.

Instead of replacing the database, we create a new database each time. Although the implementation of the bounded history encoding only requires the data of the current point in time, storing all the data brings many advantages for later analysis. The exact same data can be queried with very different queries. Each step can be repeated to check whether the results are reproducible. As mentioned above, it is necessary to query any partial period. Furthermore, the storage of all data allows a comparison of the implementation of the bounded history encoding with other established approaches.

References

[1] Stefan Borgwardt, Marcel Lippmann, and Veronika Thost. Temporalizing rewritable query languages over knowledge bases. *Journal of Web Semantics*, 33:50–70, 2015.

[2] AutoScout24 GmbH.

[3] Christopher Buhtz. Autoscout24 mining (teil 1) - webscraping mit python.

[4] Leonard Richardson. Beautiful soup documentation. April 2007.

[5] Jeff Reback, Wes McKinney, jbrockmendel, Joris Van den Bossche, Tom Augspurger, Phillip Cloud, gfyoung, Sinhrks, Adam Klein, Matthew Roeschke, Simon Hawkins, Jeff Tratner, Chang She, William Ayd, Terji Petersen, Marc Garcia, Jeremy Schendel, Andy Hayden, MomIsBestFriend, Vytautas Jancauskas, Pietro Battiston, Skipper Seabold, chris b1, h vetinari, Stephan Hoyer, Wouter Overmeire, alimcmaster1, Kaiqi Dong, Christopher Whelan, and Mortada Mehyar. pandas-dev/pandas: Pandas 1.0.3, mar 2020.

[6] Michael Bayer. Sqlalchemy. In Amy Brown and Greg Wilson, editors, *The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks*. aosabook.org, 2012.