TimeAtlas - a Time Series Data Manipulation Tool for Python

1st Frédéric Montet
iCoSys Institute
HEIA-FR
Fribourg, Switzerland
frederic.montet@hefr.ch

1st Lorenz Rychener
iCoSys Institute
HEIA-FR
Fribourg, Switzerland
lorenz.rychener@hefr.ch

2nd Jean Hennebert
iCoSys Institute
HEIA-FR
Fribourg, Switzerland
jean.hennebert@hefr.ch

3rd Jean-Philippe Bacher
Energy Institute
HEIA-FR
Fribourg, Switzerland
jean-philippe.bacher@hefr.ch

***Abstract*—With the development of TimeAtlas, a tool for time-series specific data handling, we reduce the time from research idea to prototype. In this demo, we introduce our aim and the library, showcase real-world use cases and present the roadmap of the project.**

I. Introduction

In many sectors, data collection has been an important process during the last decade. In the latter topic, the field of time series storage has seen a tremendous evolution as seen in the DB-engine ranking[1]. This is no surprise since time-indexed values are at the core of scheduling tasks, industrial processes, and many more.

Now that many industrial actors have acquired a vast amount of time series, all are eager to use this data with the latest advances in machine learning and statistical modelling. This would help to better plan and provide more economical offers to their customers and for them.

The currently available toolset in the field of time series manipulation allows for the handling of such data but is a slow and costly process. The acquisition of datasets from heterogeneous sources is always implying an unknown amount of processing time until the data is ready for modelling. Furthermore, if the task goes up to the deployment of a model in a prototyping environment, the costs are further increased.

TimeAtlas aims at providing a comprehensive toolset to go from unknown time series data loading to prototyping, including the implementation of state-of-the-art machine learning models. Such a tool would allow the industrial and research sector to quickly follow intuitions and go through trial and error phases before making the choice of a custom software development.

This demonstration introduces TimeAtlas with its aim, a presentation of the website (https://timeatlas.dev), a few scenarios inspired by real use cases and finally, a presentation of the roadmap.

II. Demonstration

A. Aim

TimeAtlas is an open source library for time series data analysis and more. It aims at providing a comprehensive API for time series analysis, prediction, as well as anomaly detection.

With such a tool, researchers and industrial actors have the possibility to follow their intuitions without spending too much time on technical intricacies.

The library is accessible for all developers levels. It is designed for researchers, data scientists and software developers.

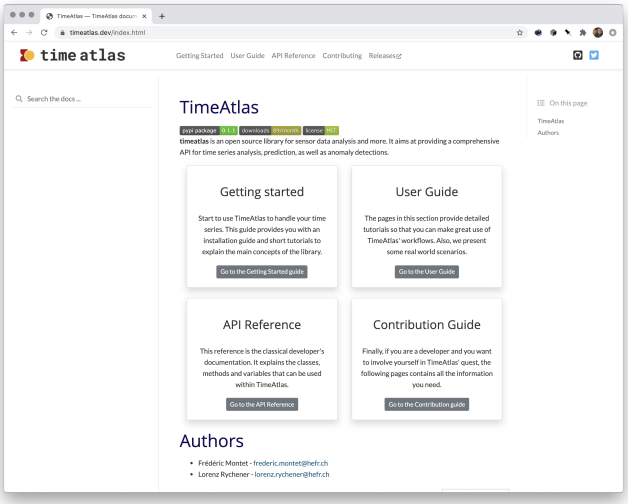

Fig. 1. Screenshot of the TimeAtlas website

B. Getting Started

As usual with Python packages, the library can be installed via the Python Package Index with the following command in 1.

```
pip install timeatlas
```

Listing 1: Installation command with Python pip

As with all time series handling library, the data can be loaded from various files such as CSV, etc. Once loaded, it is possible to get simple statistics like its minimum, maximum, different percentiles and mean.

Also, it is possible to plot the time series in various ways. TimeAtlas provides different plotting functions so that the reader's interpretation of the data can be as good as possible as seen in 3.

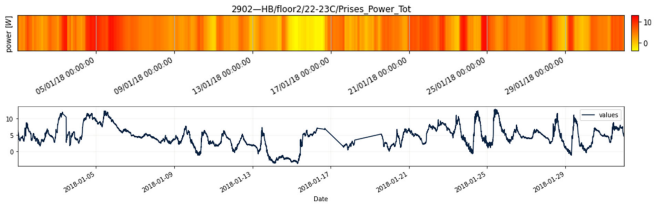

Fig. 2. Two graphical representation of the same time series

C. Use Cases

During the demonstration, three use cases will showcase real-world scenarios where TimeAtlas shines.

1) Loading data from heterogeneous sources: It often happens that data given by research partners or clients are messy and need some preprocessing before considering any other tasks. Through a step by step explanation, this use case explains the process from initial data loading of multiple time series with different indexes to the creation of a single archive containing the aggregation of the data.

2) Predict in a few lines of code: This second use case highlights the simplicity of the chosen API for model creation, fitting and prediction of a univariate and multivariate time series. It will also show how prediction results can be stored and plotted, including their confidence intervals.

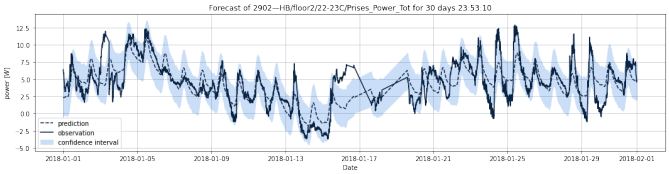

Fig. 3. Example of a time series prediction

3) Computing a Deviation from the Norm with an Unsupervised Anomaly Detector: The third use case will demonstrate, on the base of the previously obtained prediction, how to detect anomalous sections of data. For that, a method similar to MULDER[2] is used together with quantile thresholds to define three levels of warnings. The task is presented on two objects: an ice rink with temperature sensors and in an office environment with many available sensors.

D. API Reference

This part of the demonstration focuses on the structure of the different elements of the library. An introduction of the concepts of each object present in the library is given.

E. Current state and roadmap

The published package of TimeAtlas is in version 1.1. It includes the stable support of base functionalities, together with documentation and unit-tested codebase.

In the following sections are an explanation of the future releases.

1) v0.2 - Consolidation: On the base of the initial feature set established in v0.1, this second minor release focuses on the stabilisation of the overall library. It includes:

- *v0.1.1* [done] Documentation and test suite from v0.1
- *v0.1.2* [in progress] Multivariate Time Series support
- *v0.1.3* [to do] Better Visualisations
- *v0.1.4* [to do] Initial Models (LSTM, Facebook Prophet and linear regressions)
- *v0.1.5* [to do] Initial anomaly detector (Surprise based detector)
- *v0.2* [to do] Documentation and Tests from v0.1.2 to latest

2) v0.3 - Modelling, Validation and Detection: The goal of this version is to implement state-of-the-art models and detectors implementations as well as validation methods. This includes models like Temporal Convolutional Networks [3] or Transformers for time series[4].

The validation module will be implemented to evaluate the quality of the trained model and envision its possible usage in real-life scenarios. It will be inspired by the method used in the M Competitions[5]

Finally, this version includes also the definition of a generic anomaly detection API and its base implementation on the currently implemented detectors.

3) Plan for 2021: Next year, we will add two main features. The first is a time series editor. It will allow operations on time series such as the addition of trends, anomalies, clipped values and more. The goal of this feature is to accelerate research in anomaly detection and make trials with synthetic data mimicking real data features as close as possible.

The second feature is the publication feature. The latter will add the option to publish models and detectors online in the form of a REST API. The target of this feature is to allow the sharing of a predictor or anomaly detector with other actors in a very short time and cost-effective way.

III. Additional Ressources

The documentation of the project can be found at https://timeatlas.dev and its code repository at https://github.com/timeatlas-dev/timeatlas.

References

[1] solid IT GmbH. Db-engines ranking - trend of time series dbms popularity, 2020.

[2] L. von Werra, L. Tunstall, and S. Hofer. Unsupervised anomaly detection for seasonal time series. In *2019 6th Swiss Conference on Data Science (SDS)*, pages 136–137, 2019.

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.

[4] Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case, 2020.

[5] MOFC. The m5 competition, 2020.