ChronoWave
Software Requirements Specification

Table of Contents

1. The Concept of Operations (ConOps)

This section describes the reasons why this product needs to be built/the problems it is solving, as well as outlines the high level design on the system and different user use cases/workflows.

1.1. Current System or Situation

Currently, there are multiple different repositories for time series data on the internet today. The issue is that lots of these repositories either require payment for access, contain data in a wide variety of un-normalized formats, or are repositories for a specific type of time series data(i.e. don't support a wide variety of time series from different domains). In addition, the current systems do not have mechanisms for the comparison of forecast error metrics between different data scientists in the community.

1.2. Justification for a New System

These current systems make it difficult for owners of large data sets to share their data with the community as finding the right time series repository can be difficult as there is a lack of centralization. In addition, this lack of centralization makes it hard for Data Scientists and Machine Learning Engineers to find time series data sets to test their models on. The lack of metric comparisons in the current systems also makes it difficult for Data Scientists and Machine Learning Engineers to know where their models stand in comparison to the community. Creating a centralized repository for the storage of time series data and the comparison of time series forecasts will make it easier for DS/MLEs and Dataset owners to share data and results, which will foster a community of open source data sharing that will help the field progress even faster.

1.3. Operational Features of the Proposed System

The ChronoWave time series repository web application will come with five primary features. First and foremost is the ability to upload time series datasets to the open source repository for sharing with the community. Once uploading, the datasets will be made public on the website. The second core feature is the browsing and downloading of time series datasets for use in training forecasting models. The third feature is the ability for data scientists to upload forecasts that they have made for datasets stored in ChronoWave. These forecasts will be compared with hidden testing data and error metrics will be calculated and stored for the forecast. The fourth feature is the browsing of different forecasts made for a time series dataset and looking at the different metrics for different DS/MLE models. This allows DS/MLEs to see how their forecast error metrics compare with others in the community. The fifth and final feature will be the ability for an admin to delete time series datasets from the repository at their own discretion to ensure that, if for some reason a dataset is deemed invalid, it can be removed.

1.4. User Classes

There are three classes of users for the ChronoWave time series repository web application: Dataset Contributors, DS/MLEs, and Administrators.

Dataset Contributors - This user class provides time series data sets to ChronoWave for sharing among the community. The primary application feature that this user class will interact with is the ChronoWave data set upload page.

DS/MLEs - This user class both downloads datasets from ChronoWave for use when testing their forecasting models as well as provides time series forecasts for error metric calculation and comparison. The relevant application features to this user class are 1.) The time series download page, 2.) The forecast upload page, 3.) The time series forecast metrics browser page.

Administrators - This user class is in charge of managing the time series repository. The primary feature they will interact with is the admin page for deleting time series data sets from the repository.

1.5. Modes of Operation

There are three distinct modes of operation: 1.) Browsing mode, 2.) Providing mode, 3.) Modifying Mode.

Browsing Mode - This mode is when a user is browsing through the different time series and/or forecasts in the time series repository. This is a mode that will primarily be used by DS/MLEs and Administrators as DS/MLEs need to look for datasets for download as well as look for forecasts to compare their results to, and Administrators need to look through the repository to see if their are any data sets that need to be removed.

Providing Mode - This mode is when a user is providing new data/information to the ChronoWave system. This is used by Dataset Contributors and DS/MLEs as Dataset Contributors provide datasets for download and DS/MLEs provide new forecasts for analysis and storage.

Modifying Mode - This mode is when a user is modifying the current data already in the system. This mode is used exclusively by the administrator when removing data sets from the repository.

1.6. Operational Scenarios (aka “Use Cases”)

Use Case: Provide a Timeseries Dataset to ChronoWave

Brief description: This use case describes how a dataset contributors would provide a dataset and forecasting task to the ChronoWave time series repository for sharing with the community.

Actors: Dataset Contributor

Preconditions:

1. The contributor has a data set to share in either JSON, CSV, or Excel format
2. The contributor has a forecasting task associated with the data set

Steps to Complete the Task:

1. The contributor splits their data set into training and testing data
2. The contributor creates a *metadata.json* file that describes the time series set they have, along with the forecasting task
3. The contributor creates a zip file with all the associated time series files and metadata.json file
4. The contributor goes to the ChronoWave website and navigates to the ‘Upload Timeseries’ page
5. The contributors presses the ‘Upload’ button, selects there zip file for upload, then presses ‘Ok’

Postconditions:

The contributors data set is now published to the ChronoWave time series repository and is public for download by other community members

Use Case: Download a Timeseries Dataset from ChronoWave

Brief description: This use case describes how a DS/MLE can browse for and download time series data set from the ChronoWave time series repository

Actors: DS/MLE

Preconditions:

1. The DS/MLE is in need for time series data to test their forecasting model against.

Steps to Complete the Task:

1. The DS/MLE navigates to the 'Download Data' page in ChronoWave
2. The DS/MLE clicks the drop down menu that shows the different names of the time series data sets.
3. The DS/MLE selects a time series data set in order to see relevant metadata(contributors, domains, etc.)
4. The DS/MLE repeats steps 2 and 3 until they find a dataset they want to download.
5. The DS/MLE clicks the 'Download Button,' and selects their preferred file format(JSON, CSV, Excel).

Postconditions:

The DS/MLE has a zip file containing all the relevant time series data in their preferred format, along with a *forecast_info.json* file that describes the forecasting task specified for the time series data set.

Use Case: Upload a Timeseries Forecast for a Timeseries Set to ChronoWave

Brief description: This use case describes how a DS/MLE can upload a time series forecast they have created using a time series set from ChronoWave for error metric calculation and community comparison.

Actors: DS/MLE

Preconditions:

1. The DS/MLE has a time series forecast created using a time series data set from ChronoWave according to the time forecasting task specified for said data set.

Steps to Complete the Task:

1. The DS/MLE navigates to the 'Upload Forecast' page in ChronoWave
2. The DS/MLE clicks the drop down menu that shows the different names of the time series data sets.
3. The DS/MLE selects a time series data set that they used for creating their time series forecast
4. The DS/MLE fills out the page form asking for contributor names, forecast name, etc.
5. The DS/MLE clicks the 'Upload' button and selects their time series forecast(JSON, CSV, or Excel accepted)

Postconditions:

The DS/MLEs forecast is viewable on the 'Performance Metrics' page along with the calculated error metrics.

Use Case: View Timeseries Forecast Results on ChronoWave

Brief description: This use case describes how a DS/MLE can view time series forecast results of different people in the community(themselves included).

Actors: DS/MLE

Preconditions:

1. The DS/MLE wants to view error metrics for forecasts made for a specific time series set.

Steps to Complete the Task:

1. The DS/MLE navigates to the 'Performance Metrics' page in ChronoWave
2. The DS/MLE clicks the drop down menu that shows the different names of the time series data sets.
3. The DS/MLE selects a time series data set that they are interested in.
4. The DS/MLE is prevented with the error metrics and comparison plot for the selected forecast in comparison to the testing data set.

Postconditions:

The DS/MLEs is viewing the community error metrics for different forecasts.

Use Case: Delete a Timeseries Set

Brief description: This use case describes how an Administrator can delete a time series set from the ChronoWave repository.

Actors: Administrator

Preconditions:

1. The Administrator knows what time series data set they want to remove.
2. The Administrator knows the link to the admin page

Steps to Complete the Task:

1. The Administrator navigates to the private admin page.
2. The Administrator selects the time series set in question from a drop down menu
3. The Administrator clicks the delete button.

Postconditions:

The time series data set in question is deleted from the ChronoWave repository and is no longer available from the ChronoWave web application.

2. Specific Requirements

This section describes the requirements for the software in terms of Must Have features, Should Have features, and Could Have features. In addition, useability and performance requirements are outlined.

2.1. External Interfaces (Inputs and Outputs)

1. Timeseries Upload Input
    a. Must Haves

   i. Support CSV time series format(as long as they are parsable by Pandas)
   ii. Support single time series file upload
   iii. Support single time series forecasting task
   iv. Support time series data sets up to 1 megabyte in size
b. Should Haves
   i. Support JSON and Excel time series formats(as long as they are parsable by Pandas)
   ii. Support multiple time series in a time series data set
   iii. Support time series data up to 10 megabytes in size
c. Could Haves
   i. Support time series data up to 16 megabytes in size(maximum document size for MongoDB)
   ii. Support multiple time series forecasting tasks per time series set

2. Timeseries Forecast Upload Input
   a. Must Haves
      i. Support CSV time series forecast format(as long as they are parsable by Pandas)
      ii. Support single time series file upload
      iii. Support time series forecasts up to 1 megabyte in size
   b. Should Haves
      i. Support JSON and Excel time series forecasts(as long as they are parsable by Pandas)
      ii. Support time series forecasts up to 10 megabytes in size
   c. Could Haves
      i. Support time series forecasts up to 16 megabytes in size(maximum document size for MongoDB)
      ii. Support multiple forecasts per time series in a time series set

3. Timeseries Download Output
   a. Must Haves
      i. Support CSV time series download format(parsable by Pandas)
      ii. Support single time series file download
      iii. Time series forecasting task included with download file
      iv. Support time series data sets up to 1 megabyte in size
   b. Should Haves
      i. Support JSON and Excel time series formats for download(parsable by Pandas)
      ii. Support multiple time series in a time series data set download zipped as a single file
      iii. Support individual time series data files up to 10 megabytes in size
   c. Could Haves
      i. Support time series data downloads up to 16 megabytes in size(maximum document size for MongoDB)
      ii. Support multiple time series forecasting tasks in time series set download

2.2. Functions

1. Time Series Data Validity Checking
    a. Must Haves
        i. Time series data sets must specify what column is the time series column. This column should be checked for validity
        ii. metadata file describing data sets should be validity checked for proper formatting
        iii. Each non-time column should be validity checked to be a floating point integer
        iv. Bad input should be recoverable(ie should not cause a system wide crash)
    b. Should Haves
        i. Detecting of bad input should display a relevant and useful error to the data uploader
        ii. metadata file should be checked to correspond with other time series data provided in an upload before processing begins

2. Time Series Forecast Data Validity Checking
    a. Must Haves
        i. Time series forecasts must have columns that match the training dataset column names. This should be checked upon upload.
        ii. Bad input should be recoverable(ie should not cause a system wide crash)
    b. Should Haves
        i. The forecast should be checked to match with the forecasting task period and forecast point count.

2.3. Usability Requirements

1. All major tools(Upload, Download, Browse) should be reachable within one click of the home page.
2. There should be a ‘Help’ page that describes to Contributors and DS/MLEs how to use the web application
3. The application should be stylized well
4. The application should provide useful messages when internal errors occur so users can understand what is wrong and how to fix their problems

2.4. Performance Requirements

1. A Dataset Contributor should not have to wait for the data storage to complete on the back end(ie. once the data is sent to the back end, the formatting and storage of data should happen synchronously).
2. Page load times should be responsive under normal network condition(ie no extenuating circumstances in terms of bad internet connection)

3. Administrator time series deletion operations should take effect immediately after being performed.

2.5. Software System Attributes

1. The system should be reliable. That is, no internal errors should cause system wide failures, no partial writes of failed uploads should be written to the repository, and all well formatted input data should be accepted.
2. The system should be maintainable. That is, the software should be designed in such a way that minimizes the amount of code that needs to be changed when adding/removing a feature. This means highly modularized software architecture.
3. The system should be portable. That is, the software should try to minimize the number of external resources that need to be deployed in addition to the web application. Attempt to use a single database that can do metadata, timeseries, and plot image storage to minimize required resources.

3. References

Schwaber, Ken, and Jeff Sutherland. *The Scrum Guide the Definitive Guide to Scrum: The Rules of the Game*. 2020.

4. Acknowledgements

Contributors: Aleksandr Stevens, Isaac Perkins, Geoffrey Brendel, Cynthia Meneses