Enhancing Complex Event Processing with Predictive Analysis

Duc Linh Tran[1], Usama Kaleem[1], and Davi Nascimento de Paula[2]

[1]Affiliation not available
[2]TU Berlin

July 11, 2018

Abstract

Complex Event Processing (CEP) deals with collecting events from multiple sources, detecting patterns, filtering, transforming, correlating and aggregating them into complex events. Predictive analytics (PA) on the other hand deals with anticipating (forecast) the occurrence of events so to be able to react before these occur. We developed a system which effectively predicts thunderstorms by ingesting real-time streaming data and by using CEP and PA. For this we used weather data from National Oceanic and Atmospheric Administration (NOAA) and perform Machine Learning and other complex events to detect the patters as they happen.

Course title: IMPRO - Information Management Project

Supervisor: Marcela Charfuelán

Inhaltsverzeichnis

1. Introduction

Weather forecasts are important tools to provide early warning on weather extremes. Given the volatile climate conditions currently seen in our planet, understanding and noticing extreme weather conditions, such as imminent storms, tornadoes and avalanches - and their reasons and impacts - is necessary to protect vulnerable societies and to build safer living conditions ("High Performance Computing for Accurate Weather Forecast", n.d.). However, to achieve achieve high levels of accuracy, modern weather forecasting models rely on computationally intensive algorithms to analyze huge volumes of data. This process can take a long time to execute and may negatively impact the promptness of forecast (Chandrathilake et al. 2016).

Complex Event Processing (CEP) systems can help to address this problem. Research on CEP has been done since 1990s with Rapide (Luckham 1998). This information processing framework allows analyzing multiple data streams by finding patterns in realtime. CEP can correlate continous streams of data from different weather measurements sources, analyze it and detect situations according to modeled event patterns (Zhou, Simmhan, and Prasanna 2013). Complex Event Processing systems, however, are not able to consider events that did not happen yet (Christ, Krumeich, and Kempa-Liehr 2016). Data streams are received, processed and analyzed, but the values are not stored as they only serve to detect patterns.

In contrast to CEP, Predictive Analysis (PA) uses historical data to perform predictions by applying Machine Learning (ML) techniques. Also, ML allows to predict a proability of future events and to calculate estimates on the uncertainty of those predictions. As a major drawback, ML needs a huge data source to increase accuracy and performance (Christ, Krumeich, and Kempa-Liehr 2016). In addition to that, these models have low capacity to quickly adapt to changes in the data pattern. A model would have to be retrained, which can consume a lot of time, where as in CEP, only the patterns need to be changed.

Taking both advantages into account, with CEP analyzing current data in data streams for realtime detection and PA using historical data stream information to predict complex events, CEP can be transformed from a technology that rects to events to one that acts proactively (Christ, Krumeich, and Kempa-Liehr 2016),

In section 2, related work in the field of CEP will be briefly presented. Section 3 will cover the use case for our implementation, which is further explained in section 4. In section 5, an evaluation will be done.

2. Related Work

One of the first approaches in combining CEP and PA was done by Toth et al. (2010). The authors performed a deep literature review on innovative papers and performed an assessment of CEP and PA tools to analyze the most promising technologies and the main issues to be solved. As a main conclusion, the paper proposed to include PA results in the CEP Pattern. More recently, Christ et al. (2016) proposed the applying the concept of Conditional Event Occurrence Density Estimation (CEODE) to CEP. These algorithms calculates the probability distribution of the occurrence of specific events. By calculating the probability of typical event combinations, CEP patterns could be predicted and some decision could be taken in advance.

In the field of weather forecasting, Chandrathilake et al. (2016) showed how CEP and Machine Learning techniques could reduce the computational time to run complex weather forecasting algorithms. By using ML to cluster some regions and identify which had more probability of thunderstorm occurence, the authors could reduce the time needed to run specialized weather forecasting algorithms. This is a very sensitive issue in the field of meteorological forecasting, as the specialized systems are very resource consuming. Also in this field, Banchero (2017) showed the benefits of combining different data-sources and using weather convection parameters applied to ML algorithms to predict hail storms.

3. Use case

The use case presentend in this paper focus on thunderstorms predictions. This extreme weather event have a destructive power, such as high winds, large hail, flash floods and the initiation of wildfires through production of lightning (Singh et al. 2017). Meteorologists expect that stronger or more frequent weather extremes will occur in the next years. In this scenario, an improved weather prediction is essential to extend the time to prepare for extreme events, protecting lives and reducing the impact on cities infrastructure (Lubchenco and Hayes 2012).

In this scenario, both CEP and PA can be useful. CEP is helpful to combine patterns of several measurements and to detect which weather conditions can trigger a thunderstorm. To extend this capability and allow predictions in a wider time frame, PA can, in addition to CEP, be used to estimate the patterns for the next periods.

3.1. Dataset

This study comprises 14 different cities in Germany. The occurrence of thunderstorm was analyzed in the period between January 2015 and June 2018. Two different weather data sources were combined, both provided provided by the National Oceanic and Atmospheric Administration (NOAA).

Station Code	City	Latitude	Longitude
10147099999	HAMBURG	53.63	9.99
10224099999	BREMEN	53.05	8.79
10338099999	HANNOVER	52.46	9.69
10382099999	TEGEL	52.56	13.29
10416099999	DORTMUND	51.52	7.61
10469099999	LEIPZIG HALLE	51.42	12.24
10488099999	DRESDEN	51.13	13.77
10513099999	KOLN BONN	50.87	7.14
10637099999	FRANKFURT MAIN	50.03	8.54
10708099999	SAARBRUCKEN	49.22	7.11
10738099999	STUTTGART	48.69	9.22
10763099999	NURNBERG	49.5	11.08
10852099999	AUGSBURG	48.43	10.93
10866099999	MUNCHEN	48.35	11.79

Table 1: List of stations analyzed in the presented work

3.1.1 Thunderstorm convection parameters

The thunderstorm convection parameters dataset was provided by the National Centers for Environmental Prediction (NCEP) Global Forecast System (GFS). NCEP is a department of NOAA, and the dataset contains forecasts with deterministic and probabilistic estimated for the following 16 days from the date of measurement. The global data assimilation and forecasts are made daily at midnight, 6am, 12pm and 6pm UTC. ("National Centers for Environmental Prediction", n.d.) From this dataset were obtained a set of 36 variables that are necessary to calculate the indexes to predict thunderstorms. The forecast data from GFS are provided for regular squares of 0.25 degrees of longitude and latitude. (Banchero 2017)

This data is provided in GRIB (General Regularly-distributed Information in Binary form) format. As neither Python or Flink support GRIB, the data is first converted into CSV format. The converted data is then fed into the CEP engine as a data stream. The conversion is done using the software wgrib2, provided by NCEP. After that, the data was consolidated in CSV files to be sent to Flink. This process has some drawbacks, such as the overhead to convert the files to CSV and the excessive size of final files. For our observations, each GRIB file size was around 80kb. After conversion, each file achieved around 4mb.

3.1.2 Thunderstorm reports

To indentify the days where thunderstorms happend, it was used the dataset provided by the National Centers for Environmental Information (NCEI), also a department of NOAA. It contains sensor data such as temperature, pressure, humidity and wind direction that are measured at weather stations around the globe. Also it contains inputed data about the sky conditions, occurence of rain, snow, thunderstorms and other weather events. The data is provided as a CSV file, with each file corresponding to a specific station. In our use case, the stations were located at 14 different cities.

		Prediction Time	
Analysis day	Time of Analysis	12pm	6pm
Before	12pm	x	-
	6pm	x	x
Current Day	12am	x	x
	6am	-	x

Table 2: Analysis and forecasts from GFS used in the present work. Adapted from (Banchero 2015)

4. Solution architecture

4.1 General achitecture

The general architecture to implement this project consists of two main parts: - Complex Event Processing in Java with Apache Flink with Apache Kafka as a connector - Predictive Analysis in Python with h2o library for Machine Learning

As the data streams can have a huge size, Apache Flink is used to process the stream. Unlike other competitors (i.e. Spark), Flink supports realtime data streaming instead of using mini batches. Flink also supports CEP with the flink-cep library which enables the predefinition of classes and methods to set atterns.

The input and output of the two components are linked with Apache Kafka. Kafka is used for building real-time data pipelines and streaming apps. It is horizontally scalable, fault-tolerant, wicked fast, and runs in production in thousands of companies. We are using Kafka as a streaming source provider and can be used as a sink as well. It works by providing Producers and Consumers, which are implement by various frameworks such as Apache Flink, Spark or a library in Python. By doing so, different programming languages and systems can be combined, which enables us to implement CEP and PA in completely different structures. For testing purposes, a csv file containing streaming data was used. To emulate a data stream, the table will be parsed in python and used as a producer via Kafka.

4.2. Complex Event Processing Component

The input of the CEP component will be a data stream containing various events such as temperature. After being parsed into an object, it will be checked upon known pattern. Fig 1. shows three patterns to detect thunderstorms and additional properties.

TABLE I. WEATHER PREDICTIONS ACCORDING TO LI VALUE.

Lifted Index	Weather Prediction
$LI \geq 0K$	Stable atmosphere - no thunderstorms possible
$0K > LI \geq -2K$	Thunderstorms possible
$-2K > LI \geq -6K$	Thunderstorms likely
$-6K > LI$	Severe thunderstorms likely

TABLE III. WEATHER PREDICTION ACCORDING TO CIN VALUE.

CIN	Weather Prediction
-50 J/kg < CIN < 0	Weak
-51J/kg < CIN < -199 J/kg	Moderate
-200 J/kg < CIN	Strong

Figure 1: Simple Thunderstorm patterns as proposed by Chandrathilake

Based on these tables, a detection class is implemented, which will serve as the basic event. The following code excerpt shows the Flink implemention of two lifted index patterns, which result in a complex pattern.

```
Pattern<ThunderStormEvent, ?> liftedWarning = Pattern
                .<ThunderStormEvent>begin("Lifted1")
                .where(new IterativeCondition<ThunderStormEvent>() {
                    @Override
                    public boolean filter(ThunderStormEvent thunderStormEvent,
                    Context<ThunderStormEvent> context)
                    throws Exception {
                        //System.out.println("hiEv2");
                        return thunderStormEvent.getLiftedIndex() >= -2;
                    }
                })
                .next("Lifted2")
                .where(new IterativeCondition<ThunderStormEvent>() {
                    @Override
                    public boolean filter(ThunderStormEvent thunderStormEvent,
                    Context<ThunderStormEvent> context)
                    throws Exception {
                        //System.out.println("loEv2");
                        return thunderStormEvent.getLiftedIndex() <= -2;
                    }
                })
                .within(Time.seconds(60));
```

By using a window with the size of two values, two simple events can be tracked. The first one, related to the "Lifted1" begin command, triggers when the lifted index shown in Fig. 1 is above -2. This pattern check is applied to every incoming event in the data stream. The second one, which is indicated by the next command, checks whether the proceeding event has a lifted index value below -2. The complex event pattern results in the combination of the simple ones. For the CIN index, the implementation for the pattern detection is the same.

Because the result of the pattern checks is treated as a data stream, the complex event processing result can be used for another complex event processing. This feature is used to combine the CEP result with the Predictive Analysis. The prediction only is meaningful as a result, when it is providded before the CEP detects the thunderstorm. Therefore, the PA result will be used as an event.

4.3 Predictive Analysis

Thunderstorms forecasting is a tool historically used to protect lives and properties. In general, standard models identify a single behaviour a storm could potentially have, which could be later confirmed or not. Predictive analysis can increase the power of these models identifying the first signs of the probability of a thunderstorm. As a result, these models can tell emergency responders how the weather events can impact people, buildings and access points. In this way, local governments can predict damage and prevent post-event behaviors hence protecting lives and saving money ("Predictive Weather Data:

Forecasting Storm Paths to Protect", n.d.)

The predictive analysis was done using regression models to estimating the occurence of thunderstorms. The statistical models are Logistic Regression and Random Forest. Both models were implemented using Python 3.6 and the library H2O. This library was chosen because it is open source and provides interesting features, such as a parallel processing engine, analytics, math, and machine learning libraries, along with data preprocessing and evaluation tools (Landset et al. 2015).

4.3 Research goals

The present use case will predict the probability of happening a thunderstorm at 12pm during the months of January to June 2018. In 14 German cities. To perform theses predictions, the datasets of 24 hours, 18 hours and 12 hours before the predicted period were used, combined with historical data for the same period (January to June) from 2015 to 2017. The Table 1 shows the indexes forecasts used for each prediction.

With this dataset, the present study aims to answer two main research goals:

a) Compare the performance time for weather forecast predictions, based on the framework proposed by Chandratilake et al. (2016)
b) Apply machine learning and conditional density estimations to predict future patterns of CEP, based on the framework proposed by Christ et al. (2016)

To analyze the research goals, we selected some weather indexes that are generally correlated with the occurence of weather events such as thunderstorms. These indexes follow simplified conceptual models of the conditions that cause thunderstorms. Thresholds have been defined for most of the parameters to transform them to different levels of warnings (Grieser 2012).

5. Methodology and discussion

5.1 Comparing performance time for weather forecast predictions

In this first research goal, this study aimed to identify how CEP can improve the performance of analysis of weather forecast predictions by reducing the computational time. Based on the thresholds presented on Fig. 2, we used CEP results to trigger the execution of the Machine Learning models. The result of the CEP implemention is shown in the code block below. Due to many additional unknown weather factors though, which cannot be explained by us due to missing knowledge of the domain, the patterns only detected 60% of the thunderstorms. To increase these numbers, the PA component is tasked to find additional patterns for the CEP. If at least one city presented values for Convective Inhibition higher than 50J/kg and Lifted Index less than -2K, CEP triggered the ML algorithm to predict for the selected regions.

```
1> Moderate weather condition based on CIN -64.9546
1> Moderate weather condition based on CIN -52.2341
1> Moderate weather condition based on CIN -72.7248
1> Moderate weather condition based on CIN -51.2168
1> Moderate weather condition based on CIN -71.5582
```

```
1> Thunderstorm possible based on LI -2.0
1> Thunderstorm possible based on LI -2.2
1> Thunderstorm possible based on LI -9.1
1> Thunderstorm possible based on LI -2.0
1> Thunderstorm possible based on LI -4.2
```

	Isobaric surface [hPa]					Surface
Variables	300	500	700	850	925	
Geopotential Height [gpm]	x	x	x	x	x	
Relative Humidity [%]	x	x	x	x	x	
Temperature [K]	x	x	x	x	x	
u-component of wind	x	x	x	x	x	
v-component of wind	x	x	x	x	x	
Vertical velocity (pressure)	x	x	x	x	x	
Surface lifted index						x
Convective available potential energy						x
Convective inhibition						x

Table 3: Variables and indexes forecasted by GFS used in this paper. Isobaric surfaces are measured in hectopascals (hPa) (Banchero 2015)

5.1.1 Machine Learn Algorithms

To perform this analysis, we manually modeled and tested several algorithms provided by H2O library, among of them Decision Trees, Logistic Regression, Naive Bayes. We used the data from the first semester of the years 2015 to 2017 as training and validation data, and the data from January to June 2018 as test set. The selected variables were based on (Banchero 2015) and are presented on the After testing more than 5 different algorithms, none of them presented a good performance. Although the accuracy was higher than 98% (which could be considered as a very good performance), the false positive rate was around 60%. This evaluation measure is calculated as the number of incorrect positive predictions divided by the total number of negatives (Fawcett 2006). In order to identify if there was a model that suited well to the data, we used the AutoML functionality provided by H2O. It uses Stacked Ensembles and includes automatic training and tuning of many models within a defined time-limit to identify which models have an optimal predictive performance (*AutoML: Automatic Machine Learning — H2O 3.20.0.2 Documentation*, n.d.). In this case the models selected by AutoML showed a slight better performance in accuracy. The best model was a GBM algorithm and achieved an accuracy of 98.7%. But again the false positive rate was close to 50%, which cannot be considered a good result.

5.1.2 Comparison of performance with CEP

After selecting the GBM model proposed by H2O AutoML, the algorithm was applied to the regions where CEP identified the trigger values for Convective Inhibition and Lifted Index. The next step was to compare the time needed to run the algorithm. Rhe scenario of filtering the regions with CEP showed a better performance, with an average time of processing of 0.8s, while to analyze the whole dataset was around 2s. There was no significant difference in accuracy and false positive rate levels

5.2 Apply machine learning and conditional density estimations to predict future patterns of CEP

To perform this task, we were interested in predicting the values of the indexes Convective Inhibition and Surface Lifted Index, since these ones identified by CEP as triggers of thunderstorms, as presented on 1. To do that, the variables shown on 3 were used. Based on the forecasts from 24, 18 and 12 hours before, the model aimed to forecast the result of Convective Inhibition and Lifted Index. The model used was a Linear Regression, and presented a R^2 of 0.7.

6. Summary

The combination between Predictive Analysis and Complex Events Processing frameworks can present good results. Although, in the presented study the results were not as good as expected. Due to the low number of the occurrence of thunderstorms during the analyzed period, the models provided a high level of false positive rates. Also, albeit the convection parameters used with CEP (Convective inhibition and Lifted index) are useful to the identification of thunderstorms, false alarm rates are generally high and probability of detection can be low (Grieser 2012). As suggestion to future works, the presented model could be extended by attaching specialized models for detecting rare events. Furthermore, this same data and approach could be applied to the identification of most common weather events, such as raining or snow. These events are more common during certain periods of the year and can provide a more balanced dataset.

References

n.d. https://ec.europa.eu/digital-single-market/en/news/high-performance-computing-accurate-weather-forecast. https://ec.europa.eu/digital-single-market/en/news/high-performance-computing-accurate-weather-forecast.

Chandrathilake, H.M.C., H.T.S. Hewawitharana, R.S. Jayawardana, A.D.D. Viduranga, H.M.N. Dilum Bandara, Suresh Marru, and Srinath Perera. 2016. "Reducing Computational Time of Closed-Loop Weather Monitoring: A Complex Event Processing and Machine Learning Based Approach". In *2016 Moratuwa Engineering Research Conference (MERCon)*. IEEE. doi:10.1109/mercon.2016.7480119.

Luckham, David C. 1998. "Rapide: A Language and Toolset for Causal Event Modelling of Distributed System Architectures". In *Worldwide Computing and Its Applications WWCA98*, 88–96. Springer Berlin Heidelberg. doi:10.1007/3-540-64216-$1_4$2.

Zhou, Qunzhi, Yogesh Simmhan, and Viktor Prasanna. 2013. "On Using Complex Event Processing for Dynamic Demand Response Optimization in Microgrid". *ArXiv Preprint ArXiv:1311.6146*.

Christ, Maximilian, Julian Krumeich, and Andreas W Kempa-Liehr. 2016. "Integrating Predictive Analytics into Complex Event Processing by Using Conditional Density Estimations". In *Enterprise Distributed Object Computing Workshop (EDOCW), 2016 IEEE 20th International*, 1–8. IEEE.

Singh, Martin S, Zhiming Kuang, Eric D Maloney, Walter M Hannah, and Brandon O Wolding. 2017. "Increasing Potential for Intense Tropical and Subtropical Thunderstorms under Global Warming". *Proceedings of the National Academy of Sciences* 114 (44). National Acad Sciences: 11657–62.

Lubchenco, J, and J Hayes. 2012. "New Technology Allows Better Extreme Weather Forecasts". *Scientific American* 306 (5): 5.

n.d. http://www.emc.ncep.noaa.gov/GFS/doc.php. http://www.emc.ncep.noaa.gov/GFS/doc.php.

Banchero, Santiago. 2017. "Evaluación De La Capacidad De Predicción De Granizo De íNdices AtmosféRicos.". *Feria De Tesis Del CIRN. 1. 2015 03 06, 6 De Marzo Del 2015. Castelar, Buenos Aires. AR.*. Master's thesis.

———. 2015. "Evaluación De La Capacidad De Predicción De Granizo De íNdices AtmosféRicos.". *Feria De Tesis Del CIRN. 1. 2015 03 06, 6 De Marzo Del 2015. Castelar, Buenos Aires. AR.*. Master's thesis.

n.d. http://www.ibmbigdatahub.com/blog/predictive-weather-data-forecasting-storm-paths-protect-communities-0. http://www.ibmbigdatahub.com/blog/predictive-weather-data-forecasting-storm-paths-protect-communities-0.

Landset, Sara, Taghi M Khoshgoftaar, Aaron N Richter, and Tawfiq Hasanin. 2015. "A Survey of Open Source Tools for Machine Learning with Big Data in the Hadoop Ecosystem". *Journal of Big Data* 2 (1). Nature Publishing Group: 24.

Grieser, Jürgen. 2012. "Convection Parameters". June.

Fawcett, Tom. 2006. "An Introduction to ROC Analysis". *Pattern Recognition Letters* 27 (8). Elsevier: 861–74.

n.d. http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html. http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html.

Grieser, Jürgen. 2012. "Convection Parameters". June.