Complex Event Processing Synergies with Predictive Analytics

Gabriella Tóth
Dept. of Software Engineering
University of Szeged, Hungary
gtoth@inf.u-szeged.hu

Lajos Jenő Fülöp
Dept. of Software Engineering
University of Szeged, Hungary
flajos@inf.u-szeged.hu

László Vidács
RGAI, Hungarian Academy of Sciences
lac@inf.u-szeged.hu

Árpád Beszédes
Dept. of Software Engineering
University of Szeged, Hungary
beszedes@inf.u-szeged.hu

Hunor Demeter
Nokia Siemens Networks Hungary
hunor.demeter@nsn.com

Lóránt Farkas
Nokia Siemens Networks Hungary
lorant.farkas@nsn.com

ABSTRACT

For Complex Event Processing (CEP), the synergy with Predictive Analytics (PA) is a promising research direction. In this paper we focus on the inclusion of PA technologies into CEP applications. Involving PA opens a wide range of possibilities in several application fields. However, we have observed that only a few CEP solutions apply PA techniques. We extended a CEP solution with predictive capabilities, defined the key aspects of the combination of these techniques, and summarized how CEP and PA could gain from the joint solution. Our approach is demonstrated in a proof-of-concept experiment and simulation results are provided.

Categories and Subject Descriptors

C.2.4 [**Comp-Comm. Networks**]: Distributed Systems

General Terms

Theory, Reliability, Design, Experimentation

Keywords

complex event processing, predictive analytics

1. SYNERGIES OF CEP AND PA

The synergies between CEP and PA enable new use cases or applications. PA deals with different kinds of prediction (long term, short term, classification, regression, etc.), while CEP deals with detecting real-time complex events. By combining these two areas one could predict complex events in the short term future. The reason why CEP should be enhanced with PA techniques is that in many cases the reaction should be triggered earlier, even before the occurrence of the primary event. In these cases, events can be predicted by PA models incorporated to the CEP solution.

DEBS'10, July 12-15, 2010, Cambridge, UK.

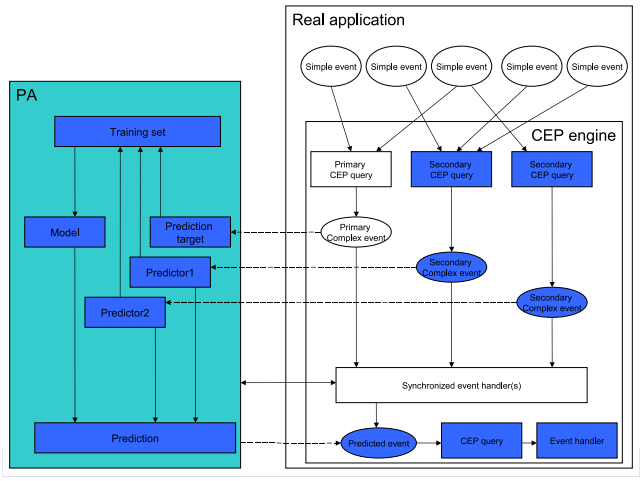

Figure 1: CEP-PA solution - logical architecture
An ellipse denotes certain simple or complex event, while a box represents an information processor component. The white boxes and ellipses are the parts of the initial CEP application, while the blue ones represent the required components and events for the CEP-PA synergy. The dashed lines represent indirect (logical) connections, while the solid lines represent physical connections.

CEP detects complex events from a system according to predefined queries based on simple events. By the connection of CEP and PA, these complex events can be predicted. The logical architecture of the interworking CEP and PA solution is illustrated in Figure 1. Let us suppose that we have an initial CEP application, represented by the white boxes and ellipses. In this system a certain complex event can be detected with the help of CEP. This event is called *primary complex event*, while the related query is called *primary CEP query*. The event is handled by the event handler, which performs an action when the event occurs, e.g. notifies the business actor or triggers an automatic reaction (ECA). The existing CEP solution is extended with the PA component in a transparent way: the CEP engine is only the interface, the original application is not affected. The PA component contains a machine learning part. A large set of predictors is necessary for building the internal machine learning model. Generally, the calculation of predictors would be the task of the original embedding application. In our case, instead of the embedding original application, the CEP engine itself

produces the predictors (an example for the two-way nature of the synergy). The CEP engine processes simple events in the original system and produces complex events based on CEP queries. These complex events are the required predictors, denoted as *secondary complex events* (SCE). The SCEs are transmitted physically by the event handler. Another requirement for PA is the target of the prediction, which is the *primary complex event* (PCE). This event can be sent to PA similarly to the SCEs. The PCEs and SCEs occur at different times, it is important to synchronize them with event handlers. Afterwards, the *training set* is extended based on the earlier defined predictors and the prediction target. The learning *model* is periodically refreshed based on the currently extended *training set*. Then PA gives a prediction for the PCE at a later time, based on the *predictors* and the actual learning *model*. This prediction is sent back to the CEP engine, which generates a new (predicted) event. By this way, PCEs can be predicted.

There are several possible extensions for the presented CEP–PA synergy. The *predicted event* can be used to define further complex events. In case of the dependency on another event which requires the avoidance of the prediction step, a new complex event based on the predicted and the controller event may help. In case of applying multiple PAs, several *predicted events* appear in the CEP engine which can be used in further CEP patterns. For example, in the case of applying 5 PA algorithms in parallel, the CEP engine gets 5 prediction events, where a complex pattern could accept the prediction if at least 3 predict the event.

2. PROOF OF CONCEPT

We performed proof of concept experiments on a CEP program which works on an entry system of a building. The task of the CEP engine is to listen the traffic data, and in case of too high *incoming* traffic it has to issue an alert (complex event) about the *overload*. With PA extension, we can not only detect the overloaded entry doors but predict as well. If the number of entrants is greater than 25 during the last 1.5 hours, the entry system is overloaded which is detected by the CEP solution. We refer to this event as the *Primary complex event*. In the current implementation, the events for the CEP engine are simulated by reading a database file [2]. The original database contains information about concrete events (e.g. conferences). However, we have not used this data, but rather we defined our custom event (incomings are greater than 25 during the last 1.5 hours) to create a CEP application. Note that the half-hour measurements could be more frequent, e.g. 10 seconds: therefore this does not affect the validity of the experiment.

The forecast is based on the number of entries and exits, and the timestamp. So, we have these simple events: the number of incoming/outgoing persons during the last half an hour and the timestamp of the measurement. The complex events are: the median, summed, maximum, minimum, average number of persons coming in during the last 2.5 hours; the difference between the entrants at 2.5 hours before and now – we had presumed that it increases when an event is approaching; the day of the week (e.g. Monday) and finally the number of people in the building. The complex events defined and detected with an Esper application [1].

The practical example contains only one SCE which represents information about all predictors. Note that certain predictors cannot be calculated with CEP query (e.g. day of the week). During the first week, the synergy does not provide any forecast, only extends the training set in the PA framework (in our solution we applied Weka [3]). At the beginning of the second week, the first learning model is built. Afterwards, the synergy can give forecasts in every half an hour, where the forecast concerns the following one hour. The training set is extended continually with the new data, while the learning model is refreshed weekly.

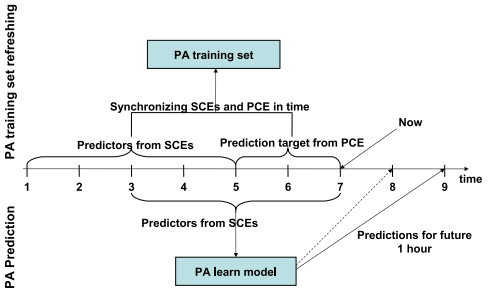

Figure 2: CEP-PA illustrated chronologically
The area above the x axis presents the extension of the training set, while the area below the x axis shows the forecasting process.

Figure 2 shows the working of proof-of-concept in time. It is a snapshot at measurement point 7. The training set is extended with data coming from measurement point 5. The predictors are calculated from SCEs on interval 1-5, while the prediction labels are calculated from PCE based on interval 5-7. By this way, the learning model based on the training set is able to forecast the events during the next one hour. In our example(Figure 2), it means the calculation of SCEs on interval 3-7, and sending these as predictors to the PA, which basically gives a forecast to the future point 9, but we accept this prediction for future point 8 as well.

Based on our results, the original CEP application is significantly improved with prediction. We detected 1017 events, from which 938 events were successfully predicted and 69 ones were false predicted. *Successful prediction* occurs if the event is predicted 0.5 or 1 hour before it really occurs. *False prediction* occurs if there was a prediction, but there are no events occurring at 0.5 or 1 hour later. The precision of the experiment was 93.15% and the recall was 92.23%.

3. CONCLUSIONS

A promising research area of CEP domain was examined: how to use PA to enhance complex event detection. A logical architecture of a PA-enhanced CEP solution was introduced. Then results on real-world data using a proof of concept implementation were presented. The achieved results are promising, the CEP solution was extended with predictive capabilities and the most events were predicted successfully.

4. ADDITIONAL AUTHORS

Additional authors: Tibor Gyimóthy (Dept. of Software Engineering, University of Szeged, `gyimi@inf.u-szeged.hu`)

5. REFERENCES

[1] Esper/NEsper. `http://www.espertech.com/`.
[2] UC Irvine Machine Learning Repository - CalIt2 building people counts. `http://archive.ics.uci.edu/ml/machine-learning-databases/event-detection/`.
[3] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2005.