Evaluation of Real-Time Traffic Applications Based on Data Stream Mining

Sandra Geisler and Christoph Quix

Abstract Traffic management today requires the analysis of a huge amount of data in real-time in order to provide current information about the traffic state or hazards to road users and traffic control authorities. Modern cars are equipped with several sensors which can produce useful data for the analysis of traffic situations. Using mobile communication technologies, such data can be integrated and aggregated from several cars which enables intelligent transportation systems (ITS) to monitor the traffic state in a large area at relatively low costs. However, processing and analyzing data poses numerous challenges for data management solutions in such systems. Real-time analysis with high accuracy and confidence is one important requirement in this context. We present a summary of our work on a comprehensive evaluation framework for data stream-based ITS. The goal of the framework is to identify appropriate configurations for ITS and to evaluate different mining methods for data analysis. The framework consists of a traffic simulation software, a data stream management system, utilizes data stream mining algorithms, and provides a flexible ontology-based component for data quality monitoring during data stream processing. The work has been done in the context of a project on Car-To-X communication using mobile communication networks. The results give some interesting insights for the setup and configuration of traffic information systems that use Car-To-X messages as primary source for deriving traffic information and also point out challenges for data stream management and data stream mining.

Keywords Data streams • Data stream mining • Data quality • Traffic management • Intelligent transportation systems

S. Geisler (✉) • C. Quix
Information Systems, RWTH Aachen University, Ahornstr. 55, 52056 Aachen, Germany
e-mail: geisler@dbis.rwth-aachen.de; quix@dbis.rwth-aachen.de

1 Introduction

When you buy a new car today, it is usually equipped with a multitude of sensors. The sensors have mainly two purposes: increasing safety and increasing comfort. Driver assistance systems, such as brake, lane, or parking assistants, are realized with these sensors. The substantial progress in making road transport safer resulted in a continuously decrease of accidents involving personal injuries and road fatalities in Germany – the number of road fatalities being lower than ever since 1950 (Statistisches Bundesamt 2011). A main goal to improve safety further, should be to not only lower the severity of accidents, but to prevent them to the greatest possible extent. This can be done by prevention mechanisms which warn road users before they approach a critical situation. To this end, communication between vehicles and their surroundings, e.g., other vehicles or traffic infrastructure, termed Car-to-X Communication (C2X), can help to exchange important safety information "sensed" by vehicles. In the Cooperative Cars (CoCar) project and its successor Cooperative Cars eXtended (CoCarX),[1] which constitutes the context of our work, a heterogeneous approach has been followed. Depending on the use case, 802.11p-based technology (pWLAN) or UMTS and its successor LTE are used for communication (cf. Fig. 1). In the CoCar system, hazard warnings are sent by vehicles to other road users. For example, when a vehicle brakes very hard, a warning message is sent to vehicles in vicinity (also called *Geocasting* or *GeoMessaging* (Fiege et al. 2011)).

Our goal in the CoCar project is to utilize the sensor information included in the messages of vehicles for deriving higher value information, such as the position of a queue-end or the current traffic state. However, there are several challenges in doing so. First, the amount of messages per time sent by vehicles from a specific area can be very high. Second, many traffic applications, especially safety applications, and their users require real-time processing and response. Data Stream Management

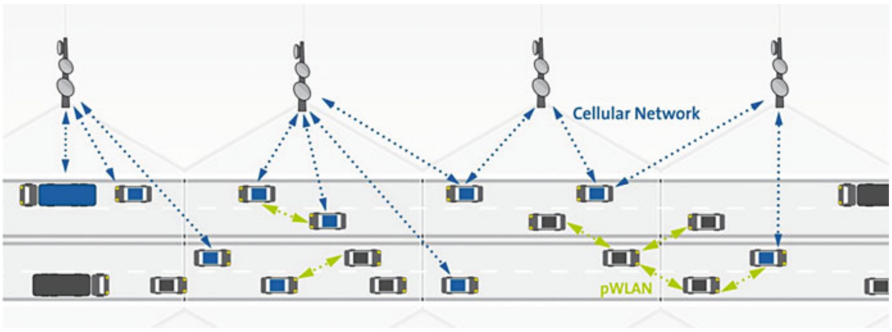

Fig. 1 The CoCarX architecture (Fiege et al. 2011)

[1] http://www.aktiv-online.org/english/aktiv-cocar.html

Systems (DSMS) and data stream mining algorithms proved to be suitable for several application fields with real-time properties. Also in traffic applications, data stream processing and mining has been applied successfully (Kargupta et al. 2010; Biem et al. 2010; Ali et al. 2010; Arasu et al. 2004), but a systematic approach for evaluating the parameters and results of such applications is still missing. Especially, the comparison of multiple data stream mining algorithms according to their suitability for various ITS applications is an interesting parameter to be analyzed systematically.

Furthermore, the quality of the processed data and the produced information is crucial. Users want to rate the reliability of information, and low-quality information should not be distributed to users to avoid desensitization and frustration. This applies not only to traffic information systems, but also to other geographic applications which rely on sensor data and apply data mining techniques to derive new information (e.g., tsunami or earthquake warning systems). The data in such applications is inherently uncertain as sensor data might be inaccurate. In addition, the methods for data analysis usually cannot achieve a 100% accuracy. Hence, the continuous quality monitoring of the processed data should be an inherent property of a data management solution for such applications.

In this paper we will present a summary of our work on a data stream-based evaluation framework for traffic applications. We first introduce the overall architecture of the evaluation framework in Sect. 2, followed by the description of the data processing and the corresponding data quality processing in Sect. 3. Subsequently, two case studies in queue-end detection and traffic state estimation are detailed in Sect. 4. The section contains also the evaluation results for various configurations of the ITS and multiple data stream mining algorithms. In Sect. 5, we will compare our approach with other works from related research areas. Finally, the lessons we learned throughout the project and some ideas for future work are discussed in Sect. 6.

2 Architecture

The evaluation framework is based on a data management and fusion architecture which was designed and implemented in the first phase of the CoCar project (Geisler et al. 2009). The main constituents of the architecture are mobile data sources which deliver the raw sensor data. Second, the data is received by a Data Stream Management System (DSMS) which processes and integrates the data. Third, to analyze the data and derive new traffic information, we integrated a data stream mining framework into the DSMS. In addition, we use a spatial database to speed up working with geospatial data. We detail these components in the following. Finally, we added a data quality monitoring component to the architecture. An overview of the architecture with its main constituents is depicted in Fig. 2.

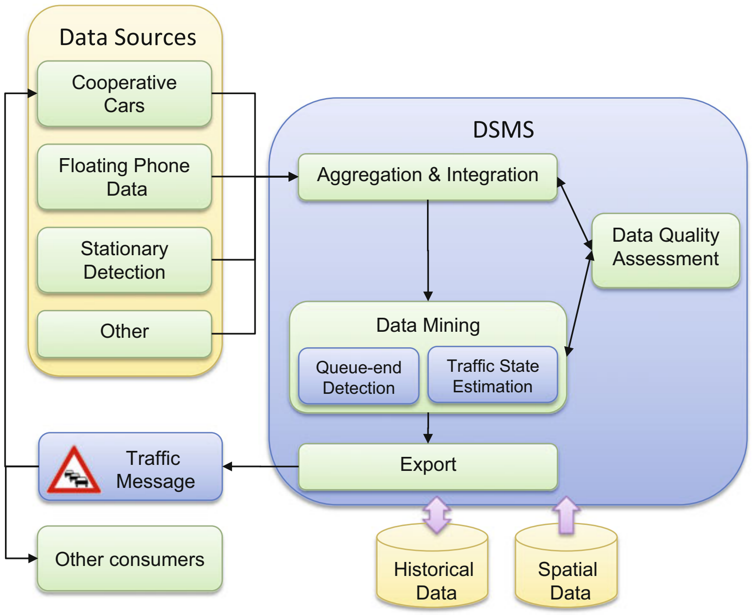

Fig. 2 The overall architecture

2.1 *Mobile Data Sources*

For reasons of reproducibility, controllability, and creation of a sufficient amount of data, we use the traffic simulation software VISSIM by PTV AG[2] to simulate the creation of data from mobile and stationary sensors. In the CoCarX project, one important data source is, of course, the CoCars sending event-based messages in case of hazards to warn other vehicles. We create CoCar messages by using the VISSIM simulation for the following events: a vehicle braking very hard and a vehicle turning on its warning flashers. For traffic state estimation we introduced another type of message, which is sent periodically by the vehicles and only contains basic information, such as position and speed. The second important data source is Floating Phone Data (FPD). FPD are anonymously collected positions of mobile phones, but the location data might be very inaccurate. Other traffic data, such as the vehicle speed, can be derived from FPD (Geisler et al. 2010). In our case studies, we focus on the CoCar messages only as we want to analyze the usefulness of CoCar messages and the influence of parameters, such as required equipment rates, for certain traffic applications.

[2]http://www.ptv.de

2.2 Data Stream Management System

We use the Global Sensor Network (GSN) system (Aberer et al. 2006) for data stream management. GSN provides a flexible, adaptable, distributable, and easy to use infrastructure. It wraps the functionality required for data stream processing and querying around existing relational database management systems, such as MySQL. The main concepts in GSN are wrappers and virtual sensors. In GSN, we provide a wrapper for each data source to receive the data. Each further processing step is encapsulated into a virtual sensor which creates the output data by a query over the input data. For data stream mining, we integrated the data stream mining framework Massive Online Analysis (MOA) (Bifet et al. 2010) as a virtual sensor into the DSMS. The DSMS distributes the derived information to consumers, e.g., as queue-end warning messages to CoCar vehicles.

2.3 Data Stream Mining

As mentioned in the previous paragraph, a virtual sensor has been built which classifies the aggregated data, i.e., each data stream element. We started off with integrating the MOA framework and all of the results presented in Sect. 4 have been made using this framework. We also made this virtual sensor more flexible, such that different kinds of data (stream) mining algorithms can be used in a convenient way. The MOA framework offers several algorithms, mainly based on Very Fast Decision Trees, a variant of the Hoeffding Tree, first proposed by Domingos and Hulten (2000). Hoeffding trees use a statistical measure called Hoeffding bound. The bound determines, how many examples are required at each node to choose a split attribute. It guarantees, that with a given probability the same split attribute is selected as the attribute which would have been chosen if all training elements were known already. The algorithm first determines the two best attributes by using a relevance measure such as information gain or the Gini index. Afterwards, if the difference between the two attributes' relevance values excels the calculated Hoeffding bound for the examples seen so far, the best attribute is chosen for the split. For classification, the MOA framework provides single learners with various voting strategies as well as ensemble algorithms and concept-adapting algorithms. For detailed descriptions of the algorithms please see Bifet and Kirkby (2009). In the mining virtual sensor, the classified stream elements are extended with an attribute for the classification result, e.g., the estimated traffic state.

2.4 Spatial Database System

Spatial database systems have been introduced to ease and speed up the work with spatial data using special data types and functions. In our architecture, we store and query the road network by using the spatial functionality of Microsoft SQL Server

2008.[3] We export the roads (or links) from the traffic simulation and store them as curve objects in the database. In the database, we divide the links into sections of equal length, e.g., sections of 100 m length each. Sections are the units of interest for our evaluations, e.g., the traffic state will be determined for each section. Each time we have to locate a vehicle on the network, the spatial database is queried to find the section the vehicle is driving on.

3 Data Stream Processing

To show the effectiveness of our framework for the purpose of a systematic evaluation of ITS applications, we set up two case studies, namely queue-end detection and traffic state estimation, detailed in Sect. 4. We will first explain the processing of the data streams throughout the DSMS as it is similar for both case studies. Subsequently, the data quality framework we implemented to measure and rate the effectiveness and efficiency of the traffic applications is elucidated.

3.1 Processing Flow

GSN provides two structures to work with data streams: wrappers and virtual sensors. The wrappers manage the connections to the data sources, receive the data and convert it to relational stream elements. In our case studies, the CoCar messages and ground truth messages (stating the real queue-end and the real traffic state) are produced by the traffic simulation and sent via TCP to the DSMS. Virtual sensors filter and process the data, e.g., aggregate and integrate data. One common issue in mobile data detection is that the measured positions contain some error whose amount depends on the used positioning technique. This error influences the accuracy of traffic information (Geisler et al. 2010). In case of the CoCar messages, we can assume GPS positioning accuracy. To approximate reality as close as possible, the exact positions in the messages created by VISSIM have to be degraded. To this end, the stream elements created from the CoCar messages are forwarded to a degradation virtual sensor. The next virtual sensor matches the positions of the CoCar messages to the road network, a process also termed Map Matching. It tries to find the section where the vehicle actually is located.

The next step is the preparation of the data for data stream mining. In our approach, we aggregate data, such as speed or acceleration, of CoCar messages coming from the same section for a certain time window. We utilize classification algorithms for data stream mining in our framework, i.e., each incoming tuple is

[3]Other products can also be used in the architecture. We implemented the same functionality also for PostgreSQL and PostGIS.

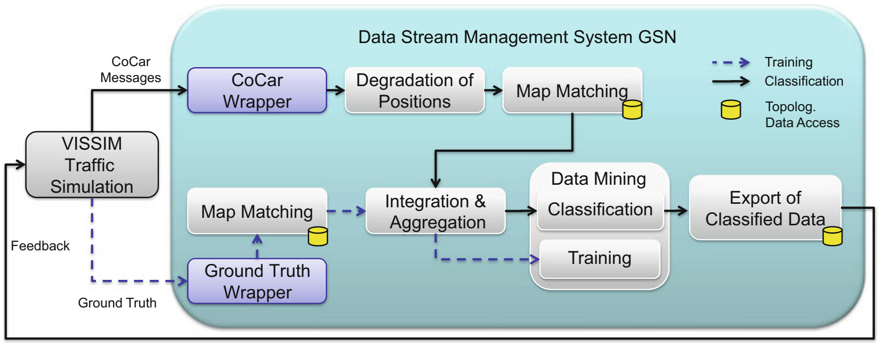

Fig. 3 Data stream processing flow

assigned to a specific class. For example, in queue-end detection either the class "Queue End" or the class "No Queue End" will be assigned to each stream element. For each run a classifier was learned from scratch. Afterwards, the classified element is distributed to consumers, e.g., sent to the traffic simulation, where it can be used to visualize the corresponding event. The message contains a confidence value, which indicates the reliability of the message based on data quality parameters calculated throughout the data processing in the DSMS. The complete data stream processing flow is depicted in Fig. 3.

3.2 *Data Quality Processing*

Data quality (DQ) plays an important role in DSMS as there is usually a trade-off between accuracy and consistency on the one hand, and timeliness and completeness on the other hand. We follow a holistic approach for DQ management in data streams which is based on a comprehensive ontology-based data quality framework. We will briefly explain the framework here, as we use it to measure the effectiveness and efficiency of the traffic applications in our experiments with it. The framework is presented in detail in Geisler et al. (2011).

Our aims in designing a DQ framework for DSMS were flexibility and adaptivity as the framework should not be restricted to traffic management applications. Even within the traffic domain, there are different ways to measure DQ depending on the data sources which are available, and the data processing steps which are applied in the DSMS. For example, to measure the correctness of a CoCar message which signals a specific event (e.g., an icy road has been detected by the car sensors), we may check whether this information is confirmed also by other cars in the same area. This can be done by computing a correctness measure while several CoCar messages are aggregated in a query in the DSMS. If an external source with weather information is available, we may also want to apply a rule to check the consistency of the information (e.g., if the temperature is above 10° C, the road cannot be icy).

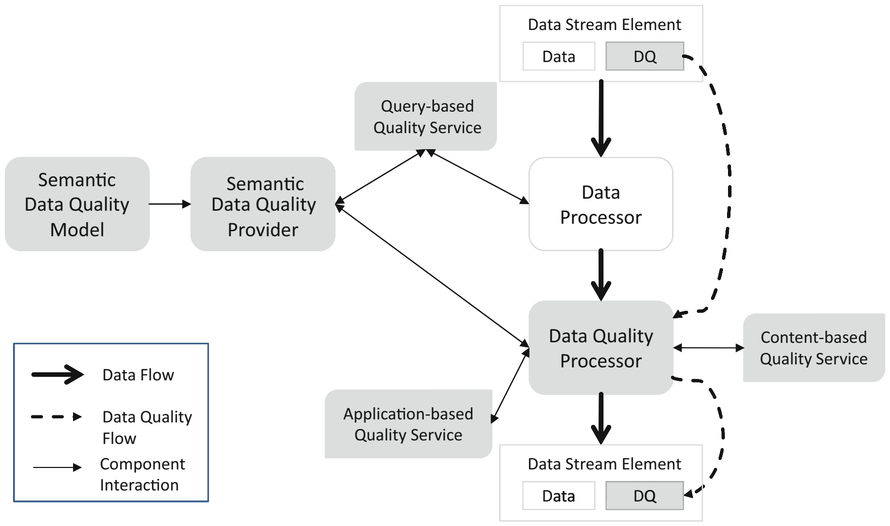
Data Stream Element
Data
DQ
Query-based Quality Service
Semantic Data Quality Model
Semantic Data Quality Provider
Data Processor
Data Quality Processor
Content-based Quality Service
Application-based Quality Service
Data Stream Element
Data
DQ
Data Flow
Data Quality Flow
Component Interaction

Fig. 4 Architecture of the ontology-based DQ component

Therefore, it is not reasonable to just select a set of DQ dimensions and DQ metrics, and hardcode them in the DSMS. A flexible and extensible DQ framework is required which allows also the definition of application-specific DQ metrics. In our approach, we distinguish three types of DQ metrics: (i) *content-based* metrics, (ii) *query-based* metrics, and (iii) *application-based* metrics. Content-based metrics use semantic rules and functions to measure data quality, e.g., as stated in the consistency check mentioned above. Query-based metrics are methods to compute the DQ for certain query operators, e.g., the accuracy of an aggregation operator is the average accuracy of the input tuples. We use a query rewriting approach to offer more flexibility than in other approaches (e.g., Abadi et al. 2005; Klein and Lehner 2009). Finally, application-based metrics measure DQ by any kind of application-specific functions, e.g., confidence values of a data mining classifier are provided by the data mining system.

Architecture of the DQ Framework

The data quality framework consists of several components, which are depicted in Fig. 4. To enable a flexible approach, all data quality related metadata is organized in a semantic data quality model, namely a data quality ontology. An ontology is a well suited tool to formalize the DQ model as it can be easily extended and adapted by users due to its human-readability and availability. The ontology includes a data stream part, which describes the concepts of data streaming, such as streams, windows, tuples, or attributes. Furthermore, concepts describing data quality, such as data quality metrics or dimensions, are incorporated in a data quality part. The data quality factor concept establishes a relationship between these parts defining which quality dimensions are measured with which metric and for which stream element in the data stream management system.

When incorporating DQ into the DSMS, the set of data stream attributes is extended by a set of DQ attributes, where each attribute represents a DQ dimension. We define these streams as *quality-affine* data streams. Syntactically, quality attributes are not distinguishable from data attributes, but semantically, they are handled differently and must be recognizable during data processing.

The three different types of DQ metrics are supported in the framework by corresponding *quality services*. The *Semantic Data Quality Model* is the DQ ontology presented before. The *Semantic Quality Provider* acts as an interface to this ontology. The *DQ Processor* invokes the *Content-based* and *Application-based Quality Services* as required by the definitions in the ontology. The *Query-Based Quality Service* interacts directly with the *Semantic Data Quality Provider* and the data processor, as queries have to be modified in the data processing steps. In Fig. 4 all grey boxes are DQ components and are added as additional components to the DSMS.

To take into account DQ processing, the semantic descriptions for the DQ assessment are loaded from the ontology at system startup. Based on this information, the virtual sensor configurations are rewritten recursively, including the queries and the output structure of each virtual sensor in the dependency graph.

We have used the DQ component in our case studies to measure important means to rate the effectiveness and efficiency of a traffic application. For example, we measured the data mining accuracy for varying parameters and the processing time of the stream elements in the system. For the case studies described in Sect. 4, additional quality factors have been defined which are combined into a final DQ value that represents the confidence in the derived information (how reliable is the information, that a queue end has been detected on a specific section?). We showed in Geisler et al. (2011), that the quality values and confidence values produced by the data quality component indeed indicate a drop or increase in data quality (e.g., the accuracy of the Map Matching using mobile phone positions is lower than using GPS positions). We could also prove that the data quality component produces only a slight overhead in performance, memory, and CPU consumption (Geisler et al. 2011).

4 Case Studies

To show the effectiveness of our framework for the purpose of systematic evaluation of ITS applications, we set up two case studies. In this section, we summarize the experiments and results of the queue-end detection and the traffic state estimation use cases. Further details about the two case studies and results can be found in Geisler et al. (2012).

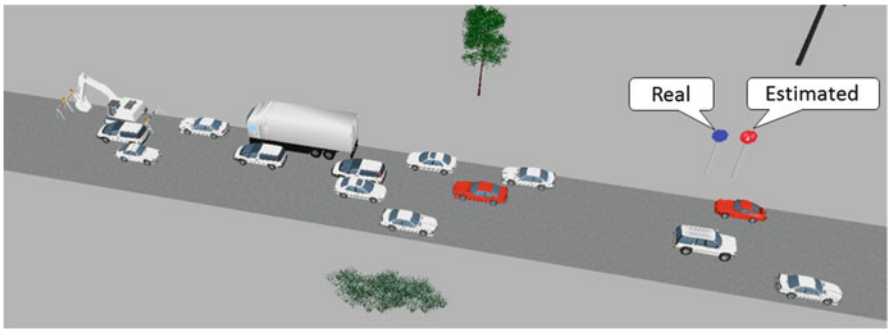

Fig. 5 Real and estimated queue-ends in the traffic simulation. The *dark gray vehicles* send a CoCar message in this time step

4.1 Queue-End Detection

The queue-end detection scenario aims at the detection of hazards caused by traffic congestion. The end of a queue constitutes a threat, as drivers may not be prepared to suddenly approach a static obstacle. The aim of our experiments was to investigate the influence of multiple parameters on the detection accuracy. For each section of the roads in the network it is determined, if it contains a queue-end or not. In this case study, traffic has been simulated on a simple road network consisting of a highway of 5–10 km length, i.e., a road with two directions with two lanes each.

One direction contains a hazard (a construction site with an excavator) narrowing the street to one lane. Depending on the traffic volume, the hazard causes a congestion. The traffic simulation sends the CoCar messages produced during simulation runs and periodic information about the ground truth (i.e., positions of queue-ends) to the DSMS. The DSMS processes this information as described in the previous section. The mining algorithm first decides, based on the received information for one section, if it contains a queue-end (classification) and learns, in which cases a queue-end is present in a section (training). If the classification identified a queue-end, a hazard warning is sent to the traffic simulation and is visualized by a red (estimated position) traffic sign as depicted in Fig. 5. The blue traffic sign denotes the queue-end as estimated by the traffic simulation (based on the maximum queue length in the last 10 s). In our simulation the queue-end warnings were located in the middle of the corresponding section. In a real-world scenario this should be set to the end of the section for safety reasons.

We tested the influence of several parameters on the output of the classification. These parameters can be divided into two categories: parameters inherent to the traffic situation and parameters inherent to data processing. Traffic situation parameters include the percentage of Cooperative Cars in the overall vehicles (penetration rate, default value: 5%), the traffic volume (vehicles per hour, default value: 3,000 veh/h), and the section size (default value: 100 m). In each simulation run (excluding only the runs with the variation of traffic volume), exactly the same traffic situation is reproduced; thus, the results of different runs are comparable. Data processing parameters comprise the size of the sliding time window for data aggregation (default value: 120 s), the mining algorithm (default algorithm: Hoeffding Tree), and the ratio of positive (section contains queue-end) and negative (section contains no queue-end) training examples presented to the mining algorithm (default option: as ratio appears, no balancing of examples). When investigating one of the before mentioned six parameters, only this parameter has been varied, all other parameters kept their default value.

For each experiment consisting of one simulation run, the following measures have been recorded: the overall accuracy (ratio of correctly classified elements to all classified elements), the sensitivity (or recall, ratio of correctly determined positive examples to all positive examples), precision (ratio of all correctly determined positive examples to the sum of all correct positive and all false positive examples), and the specificity (ratio of correctly determined negative examples to all negative examples). We recorded and plotted these measures (y-axis, each in percentage) over simulation time (x-axis, in seconds) to rate the learning progress of the algorithms during the simulation runs. If not otherwise stated, the algorithms build models from scratch in each run.

For this case study, each stream element produced by the aggregation sensor and processed for training by a mining algorithm, has the following structure:

$$\begin{aligned}&\mathit{MiningElement}(\mathit{Timed}, \mathit{AvgSpeed}, \mathit{AvgAccel}, \mathit{HasQueueEnd},\\&\qquad \mathit{WLANo}, \mathit{EBLNo}, \mathit{LinkID}, \mathit{SectionID})\end{aligned}$$

where `Timed` is the timestamp assigned by the aggregation sensor as creation time of the stream element, `AvgSpeed` is the average of speeds from the CoCar messages of the last time window (e.g., the last 120 s) for the section and link at hand, `AvgAccel` is the average acceleration from the CoCar messages for the same window, section and link, `HasQueueEnd` is the ground truth for this section (does it contain a queue-end or not in the traffic simulation?), `WLANo` is the number of messages of the type warning light announcement, `EBLNo` the number of emergency braking light messages for this section and time window, `LinkID` is the id of the link, and finally `SectionID` is the id of the section which the data was aggregated for. An example stream element would be:

$$(1338981779163, 19.5, -6.75, 1, 0, 3, 21, 2)$$

The output stream elements of the mining sensor have almost the same structure, containing additional fields for data quality, such as the mining accuracy or the timeliness, and of course the class estimated by the algorithm:

$$\begin{aligned}&\mathit{MiningElement}(\mathit{Timed}, \mathit{AvgSpeed}, \mathit{AvgAccel}, \mathit{HasQueueEnd},\\&\qquad \mathit{WLANo}, \mathit{EBLNo}, \mathit{LinkID}, \mathit{SectionID},\\&\qquad \mathit{ClassQueueEnd}, \mathit{DQ\_Timeliness}, \mathit{DQ\_MiningAccuracy})\end{aligned}$$

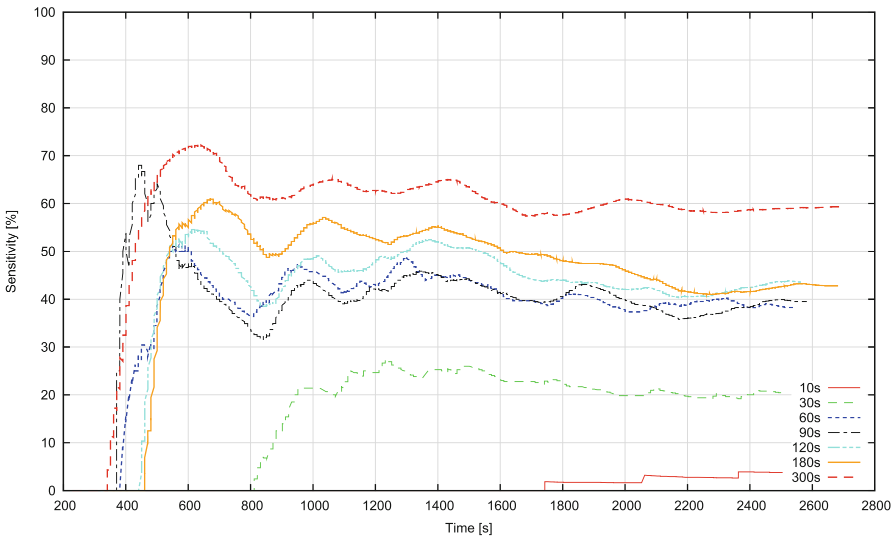

Fig. 6 Sensitivity results for varying window sizes over simulation time

We identified a set of default values for all parameters, and while one parameter was varied in experiments, all other parameters kept their default value. In most experiments, the mining accuracy was dominated by the specificity, as we had far more negative than positive examples.

In the experiments with varying window sizes between 10 and 300 s, we identified a default window size of 120 s as a good compromise for getting acceptable results for sensitivity (about 51%) as well as for specificity (about 92%), and precision (about 61%). The results for sensitivity are depicted in Fig. 6. Simulation runs with varying traffic volumes ranging from 2,000 to 4,000 showed, that, though the number of messages increases and the sensitivity is better for higher traffic volumes, it always deteriorates in the end. This is due to the fact that the ratio of positives to negatives gets highly imbalanced (ranging from 14% in 2,000 veh/h run to 9% in 4,000 veh/h run), because there are many sections that do not include a queue-end but deliver many messages. Precision on the other hand, seems to be best for 3,000 veh/h, but it does not show a clear trend for increasing or decreasing traffic volumes.

For the test with multiple penetration rates (from 1 to 10%, cf. Fig. 7) we found out, that the accuracy is not substantially influenced by the penetration rate, although higher penetration rates (from 5% on) deliver a slightly better sensitivity than the lower values. In contrast, precision increases with increasing penetration rates. For 10% penetration rate, precision gets up to 88% precision as shown in Fig. 8.

The results for simulation runs with section lengths between 30 and 300 m showed that the sensitivity increases with increasing section length. However, more tests have to be done to see if it reaches a maximum, where it decreases again. For Precision on the other hand, the best values are reached for 150 m with no clear trend with increasing or decreasing section lengths (30 m being the worst).

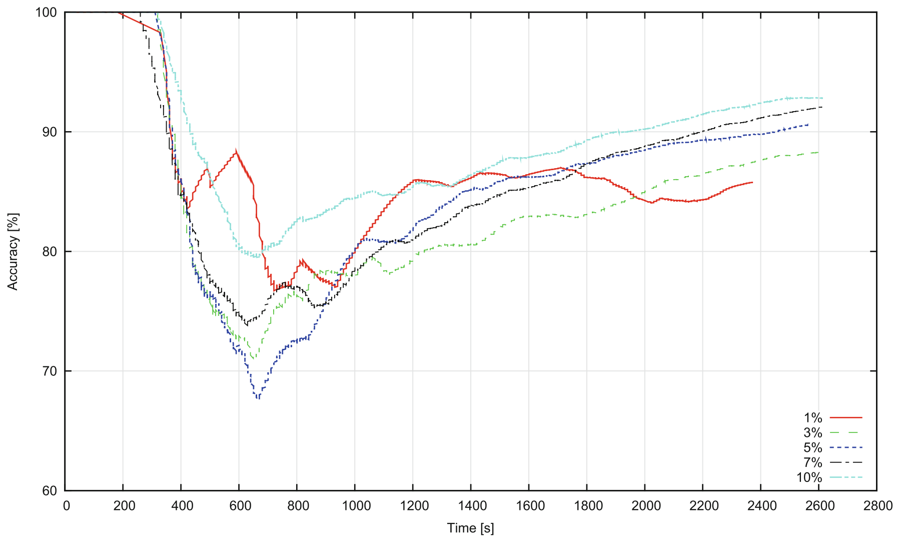

Fig. 7 Accuracy results for varying penetration rates over simulation time

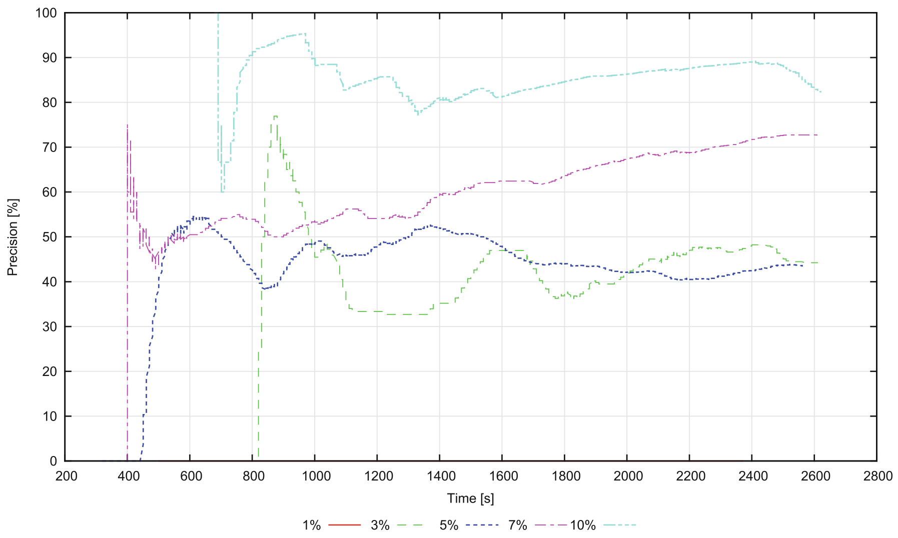

Fig. 8 Precision for varying penetration rates over simulation time

To tackle the problem of the unbalanced ratio between positive and negative examples, we also made tests using an undersampling technique in the data mining process. It drops negative examples randomly; we experimented with percentages between 0 and 90%. The sensitivity increases with increasing dropping rate as can

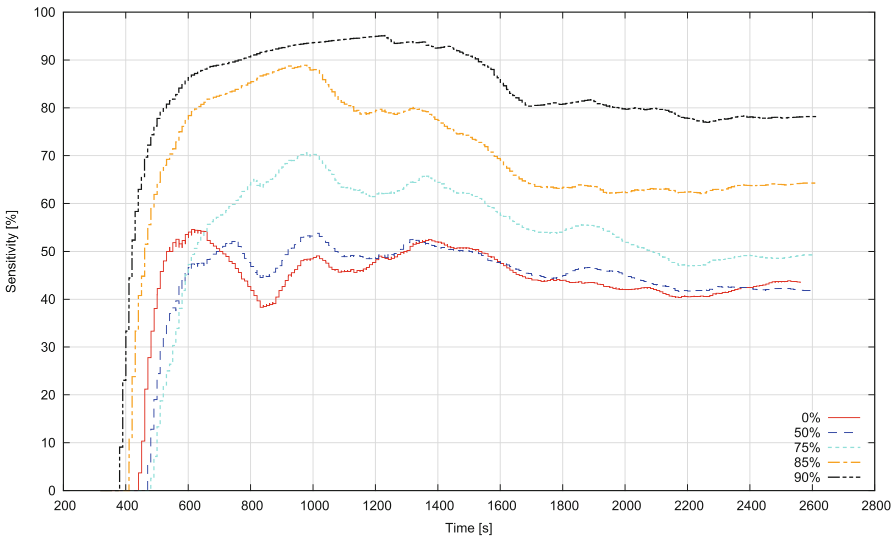

Fig. 9 Sensitivity for multiple undersampling rates over simulation time

be seen in Fig. 9, but in turn specificity and precision decrease. A good compromise could be found with a dropping rate of 85%, where sensitivity lies around 65%, specificity at about 82%, and precision at about 38%.

4.2 Traffic State Estimation

After first experiments with the queue-end detection scenario, we wanted to show the adaptability of the evaluation framework to other applications and to more complex road networks. A simple but popular traffic application is traffic state estimation. For this use case, we also used only CoCar messages. We created an artificial road network consisting of four links with two lanes each, which covers an area of approximately 4 km by 8 km. We also applied the scenario to realistic maps as shown in Fig. 10 (a part of the road network close to Düsseldorf, Germany). Another important purpose of this scenario is to compare the suitability of different stream mining algorithms. We also used data stream classification algorithms in the experiments: data stream elements are classified into one of four traffic states (`free`, `dense`, `slow-moving`, and `congested` as proposed in BASt (1999)). The ground truth for each section is calculated in the traffic simulation using the corresponding rules defined in BASt (1999). To evaluate the results, we compiled a confusion matrix which indicates the combinations of real classes and estimated classes (e.g., number of elements whose real class is dense, but have been classified as free). Based on the confusion matrix the mining accuracy has been calculated. To show the representativeness of our results, we made three runs varying only the seed

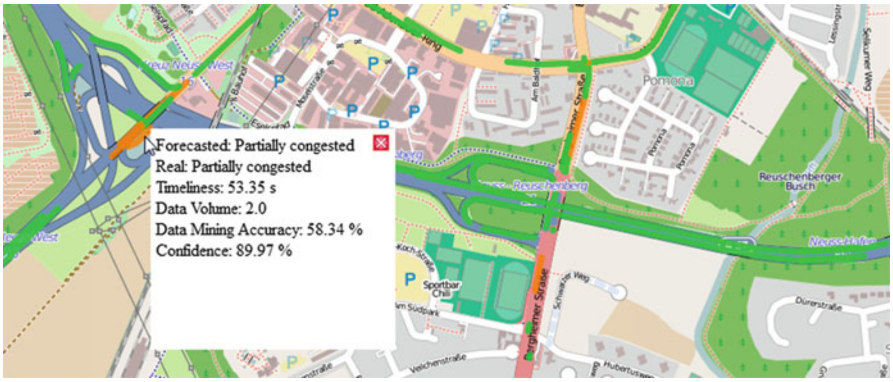
Forecasted: Partially congested
Real: Partially congested
Timeliness: 53.35 s
Data Volume: 2.0
Data Mining Accuracy: 58.34 %
Confidence: 89.97 %
Pomona
Reuschenberger Busch

Fig. 10 Visualization of the traffic state estimation scenario with OpenStreetMap

for random functions in the traffic simulation (e.g., used for vehicle speeds, entrance of vehicles in the simulation and so on). The comparison of these runs showed, that the overall accuracy of the mining algorithm only varied about 3–5% max.

The results are visualized on a map, where the estimated traffic state for each section is color-coded as shown in Fig. 10.

In the simulation runs we also used a set of default values for the variable parameters and a default classifier. We used 400 m for the section length, a window size of 60 s, and a 10% CoCar penetration rate as default values. The concept-adapting Hoeffding Option Tree with Naïve Bayes prediction strategy was used as the default classifier as it delivered the best results in comparison to other algorithms. We used no balancing algorithm in all the experiments. The input stream elements for the mining algorithms look similar to the queue-end detection scenario – only the ground truth attribute differed, containing the traffic states calculated in the traffic simulation as described. The output stream elements contain besides the additional estimated traffic state class, the timeliness, and the number of elements classified into each of the confusion matrix cells.

We first only used the emergency braking and warning light announcement messages for queue-end detection. However, we noticed that all traffic states except for `congested` were not detected well, since the event messages are only created in crowded traffic situations. Hence, we introduced periodically sent messages in the simulation which only contained basic non-event data (mainly position, speed, and acceleration). This improved the accuracy of the other classes, but still did not lead to good results, especially for the class `dense`. One problem seemed to be again the imbalanced number of examples for each class. Therefore, multiple traffic volumes on the different roads have been used in the traffic simulation to create training examples distributed over the classes (between 2,500 and 3,500 veh/h). Another improvement was the inclusion of the number of periodic messages as input parameter into the classified stream elements, which we assumed to be an indicator for the traffic density. This led to a substantial increase for the classes

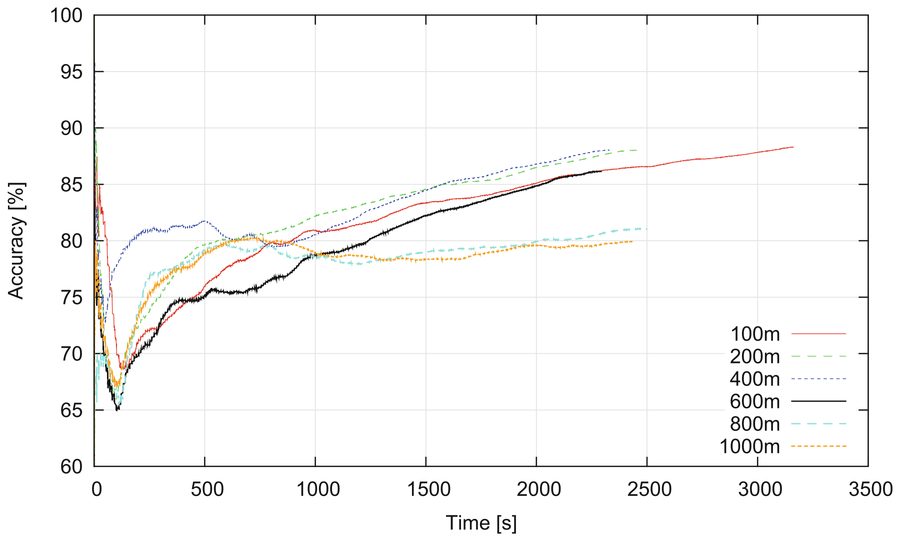

Fig. 11 Accuracy for varying section lengths over simulation time

`free`, `dense`, and `slow-moving`. Experiments with varying section lengths revealed, that sections with sizes of 100 m and over 600 m produce less accurate results than sizes of 200 and 400 m (around 87% overall accuracy). Smaller sections may not contain enough messages to indicate the traffic state, while larger sections can include more than one traffic state as depicted in Fig. 11. Hence, we used 400 m as a default value for the section length in further tests.

Similar to the queue-end detection scenario tests with varying penetration rates showed no substantial influence on the overall accuracy.

The comparison of the data stream mining algorithms of the MOA framework (Bifet et al. 2010) was divided into several experiments. First, we were interested in the influence of different voting strategies with the same classifier (in this case the basic Hoeffing Tree). It turned out that on average, the Naïve Bayes and the Adaptive Hybrid strategy were very similar in their results, but both outperformed the Majority Voting strategy. In the next group of experiments, the ensemble mining algorithms of the MOA framework were compared using the Hoeffding Tree as a base learner. The boosting algorithm (Oza 2005) was slightly better than the other algorithms with accuracies of up to 92%. However, it delivers also only a slightly better accuracy than the best single classifier algorithm in average. The results are summarized in Fig. 12.

Finally, also algorithms which are able to adapt to concept drift have been compared to see, if they can compete with the other algorithms (though our scenario does not contain concept drifts). The experiments showed, that none of them outperforms the ensemble algorithms, in fact the online boosting algorithm (Oza 2005) has proven to be slightly better than the concept drift-adapting variants.

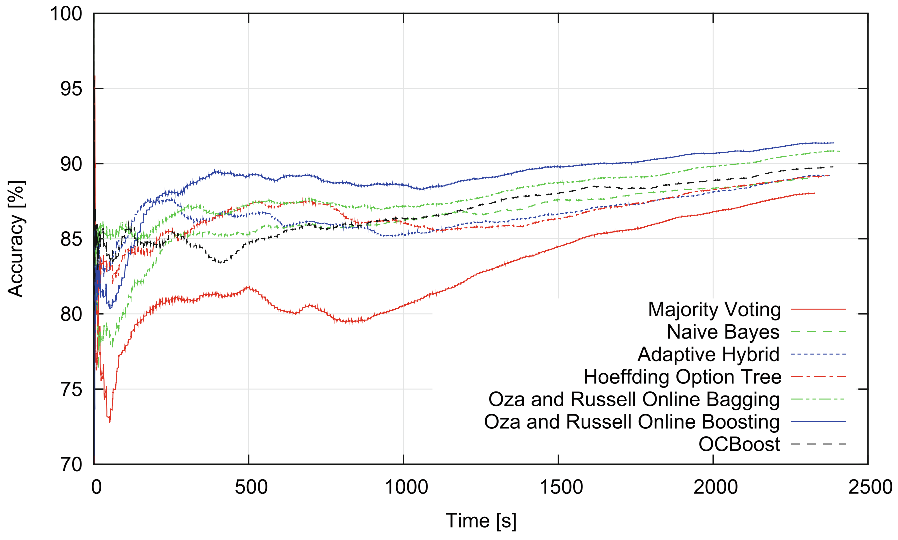

Fig. 12 Accuracy for various prediction strategies and ensemble algorithms over simulation time

Performance of data processing is a crucial aspect in data stream management systems as a huge amount of data has to be processed in a short time. Hence, we measured the processing time of the data stream elements for each sensor using the timeliness dimension. This means, that for each CoCar message the time between its creation in the traffic simulation and the processing time in the sensor is calculated. When data is aggregated over a window, the timeliness of the oldest stream element in the window is used as the timeliness value. Figure 13 shows the timeliness over simulation time for the Düsseldorf case study smoothed with a Bézier function. In this performance experiment in a 40 min simulation run, about 19,000 CoCar messages have been processed, aggregated to nearly 50,000 data stream elements which have been subsequently mined by a data stream mining algorithm. It can be seen from Fig. 13 that the sensors before aggregation only need a few milliseconds to process the stream elements, where the processing time is slowly increasing with simulation time. In the aggregation sensor, data of the last 120 s has been aggregated, hence the timeliness values are higher per se.

5 Related Work

Our work is related to a multitude of areas. DSMS have already been used to realize real-time traffic applications (Biem et al. 2010; Ali et al. 2010; Arasu et al. 2004; Kargupta et al. 2010), but none of them provided a thorough analysis of the applications, their efficiency, and the influencing factors to derive advices

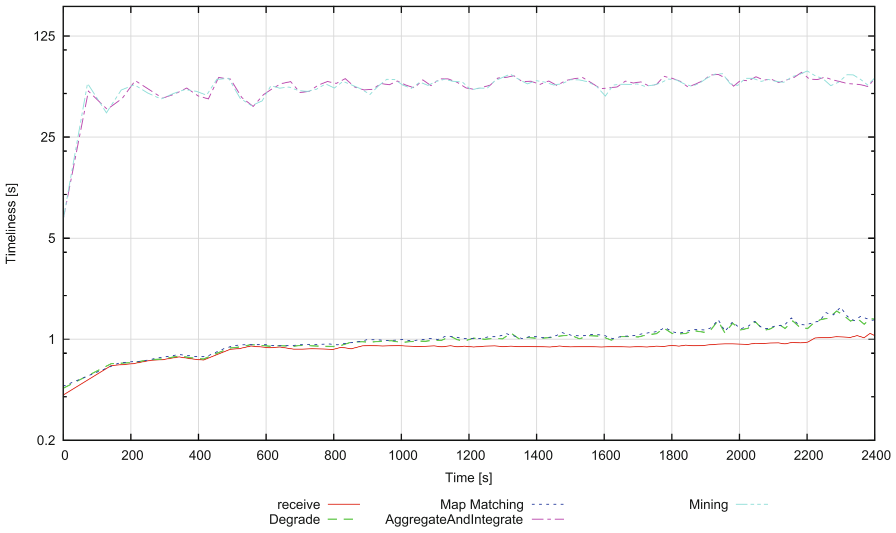

Fig. 13 Timeliness of stream elements measured over simulation time

for traffic information providers. An evaluation framework for traffic applications based on VISSIM has been proposed for example by Fontaine and Smith (2007). They investigated the influence of different parameters in the road network and mobile network on the accuracy of traffic information, such as link speed. However, they do not consider the characteristics of the processing information system and do not incorporate analysis techniques, such as data mining, to derive events or traffic information from the collected data. In our work, we investigate the influence of varying road and traffic parameters, and we also take into account the special features of the DSMS (e.g., window size, sliding step) and of the stream mining algorithms, such as concept drift. For queue-end detection, Khan presents a simulation study also based on VISSIM using stationary sensors as data source (Khan 2007). The data is analyzed using Artificial Neural Network models, predicting the current queue length based on accumulated numbers of cars and trucks at fixed locations. Although they postulate real-time processing in their information system, the used algorithm is not capable of online learning, i.e., it cannot adapt to concept changes, which can be achieved by using data stream mining algorithms as proposed in our approach. Neural networks have also been used for traffic state estimation and short-term prediction (Bogenberger et al. 2003; Chen and Grant-Muller 2001). As neural networks are easily applicable for data streams, we will also compare some of them with the performance of the decision tree algorithms included in MOA in future work.

Approaches for data quality extensions in DSMS mainly consider Quality of Service aspects, i.e., to monitor the system performance during query processing and adapt the system configuration if certain quality criteria are not met. Some of these approaches have a very coarse granularity and are only considering the quality of the outputs (e.g., Schmidt et al. 2004; Abadi et al. 2003), while others measure data quality at each operator and for each stream element or window (Abadi et al. 2005; Klein and Lehner 2009). Most approaches allow system-based quality measures, such as throughput or performance, while some also provide user-defined and content-related quality measurement beyond Quality-of-Service applications, e.g. in Klein and Lehner (2009). In contrast, we allow a very flexible and fine-granular definition of data quality and its measurement in our framework. The framework is based on a modularized ontology which allows for defining system-based as well as application-based data quality metrics using rules and numerical expressions (Geisler et al. 2011).

6 Conclusion and Outlook

For intelligent transportation systems, and especially safety applications, high effectiveness is a crucial success factor. Applications producing low quality results are quickly rejected by users, because they do not trust them or are desensitized by false alarms. The presented framework is a step towards evaluating traffic applications based on mobile and static data sources in a realistic setting. The framework enables ITS providers to test several system configurations, data sources, and algorithms and allows to derive advices for productive systems. For example, we showed, that data from C2X messages can be utilized to derive hazard warnings even if the penetration rate is low. Of course, simulation studies always have to be validated against reality and one future aim is to try out the optimal set of parameter values in a real traffic scenario. For this, a model has to be learned in several simulation runs with varying traffic situations. The model can then be used for classification of data produced by real traffic situations. Depending on the effectiveness, the model has to be adapted in simulation again. To approximate reality in the simulation as close as possible, the real traffic data (e.g., speed measures) can be used in the traffic simulation.

Our work also revealed, that ITS are an interesting application area for data stream concepts and management systems. As current and future traffic applications and their users require real-time processing and response, data stream management and mining concepts proved to be a perfect match in the case studies we analyzed and look promising for other applications. Additionally, the continuous quality monitoring in the framework helps to keep up user trust and gives feedback to ITS developers. So far, we utilized existing processing and analysis algorithms to realize the described use cases, but the experiments also showed that the potential for tweaking has not been fully exploited. For example, we will investigate more complex techniques better suited for data streams to balance training examples. Example scenarios for concept drift, e.g., to simulate a full day with bursty and low traffic periods, would be also a challenge.

Acknowledgements This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the grant 01BU0915 (Project Cooperative Cars eXtended, http://www.aktiv-online.org/english/aktiv-cocar.html) and by the Research Cluster on Ultra High-Speed Mobile Information and Communication UMIC (http://www.umic.rwth-aachen.de). We thank the PTV AG for kindly providing us with a VISSIM license. We also thank the reviewers for their valuable comments.

References

Abadi DJ, Carney D, Çetintemel U, Cherniack M, Convey C, Lee S, Stonebraker M, Tatbul N, Zdonik SB (2003) Aurora: a new model and architecture for data stream management. VLDB J 12(2):120–139

Abadi DJ, Ahmad Y, Balazinska M, Çetintemel U, Cherniack M, Hwang JH, Lindner W, Maskey A, Rasin A, Ryvkina E, Tatbul N, Xing Y, Zdonik SB (2005) The design of the Borealis stream processing engine. In: Proceedings of the CIDR, Asilomar, pp 277–289

Aberer K, Hauswirth M, Salehi A (2006) A middleware for fast and flexible sensor network deployment. In: Proceedings of the VLDB'06, Seoul, pp 1199–1202

Ali MH, Chandramouli B, Raman BS, Katibah E (2010) Spatio-temporal stream processing in Microsoft StreamInsight. IEEE Data Eng Bull 33(2):69–74

Arasu A, Cherniack M, Galvez E, Maier D, Maskey A, Ryvkina E, Stonebraker M, Tibbetts R (2004) Linear road: a stream data management benchmark. In: Nascimento MA, Özsu MT, Kossmann D, Miller RJ, Blakeley JA, Schiefer KB (eds) Proceedings of the 30th international conference on very large data bases (VLDB), Toronto. Morgan Kaufmann, pp 480–491

BASt (1999) Merkblatt für die Ausstattung von Verkehrsrechnerzentralen und Unterzentralen (MARZ). Bundesanstalt für Straßenwesen. (in German)

Biem A, Bouillet E, Feng H, Ranganathan A, Riabov A, Verscheure O, Koutsopoulos HN, Moran C (2010) IBM InfoSphere streams for scalable, real-time, intelligent transportation services. In: Elmagarmid AK, Agrawal D (eds) Proceedings of the ACM international conference on management of data (SIGMOD), Indianapolis. ACM, pp 1093–1104

Bifet A, Kirkby R (2009) Data stream mining – a practical approach. University of Waikato. http://www.cs.waikato.ac.nz/~abifet/MOA/StreamMining.pdf

Bifet A, Holmes G, Kirkby R, Pfahringer B (2010) MOA: massive online analysis. J Mach Learn Res 11:1601–1604

Bogenberger K, Belzner H, Kates R (2003) Ein hybrides Modell basierend auf einem Neuronalen Netz und einem ARIMA-Zeitreihenmodell zur Prognose lokaler Verkehrskenngrößen. Straßenverkehrstechnik 47(1):5–12. (in German)

Chen H, Grant-Muller S (2001) Use of sequential learning for short-term traffic flow forecasting. Transp Res Part C Emerg Technol 9(5):319–336

Domingos P, Hulten G (2000) Mining high-speed data streams. In: Proceedings of the 6th ACM SIGKDD international conference on knowledge discovery and data mining, Boston. ACM, pp 71–80

Fiege G, Gasser T, Gehlen G, Geisler S, Jodlauk G, Phan MA, Quix C, Rembarz R, Wiecker M, Westhoff D (2011) ITS services and communication architecture. Deliverable D03, Cooperative Cars eXtended

Fontaine MD, Smith BL (2007) Investigation of the performance of wireless location technology-based traffic monitoring systems. J Transp Eng 133:157–165

Geisler S, Quix C, Gehlen GG, Jodlauk G (2009) A quality- and priority-based traffic information fusion architecture. Proc. of the 16th World Congress on intelligent transport systems and services (ITS), Stockholm, Sweden

Geisler S, Chen Y, Quix C, Gehlen G (2010) Accuracy assessment for traffic information derived from floating phone data. Proc. of the 16th World Congress on intelligent transport systems and services, Busan, Korea

Geisler S, Weber S, Quix C (2011) An ontology-based data quality framework for data stream applications. In: Proceedings of the ICIQ, Adelaide

Geisler S, Quix C, Schiffer S, Jarke M (2012) An evaluation framework for traffic information systems based on data streams. Transp Res Part C 23:29–55

Kargupta H, Sarkar K, Gilligan M (2010) MineFleet®: an overview of a widely adopted distributed vehicle performance data mining system. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining, Washington. ACM, pp 37–46

Khan A (2007) Intelligent Infrastructure-based queue-end warning system for avoiding rear impacts. IET Intell Transp Syst 1:138–143

Klein A, Lehner W (2009) Representing data quality in sensor data streaming environments. ACM J Data Inf Qual 1(2):1–28

Oza N (2005) Online bagging and boosting. Proceedings of the IEEE International conference on systems, man and cybernetics, Waikoloa, Hawaii, USA, pp 2340–2345. IEEE. http://dx.doi.org/10.1109/ICSMC.2005.1571498

Schmidt S, Berthold H, Lehner W (2004) QStream: deterministic querying of data streams. In: Nascimento MA, Özsu MT, Kossmann D, Miller RJ, Blakeley JA, Schiefer KB (eds) Proceedings of the thirtieth International conference on very large data bases (VLDB), Toronto. Morgan Kaufmann, San Francisco, Toronto, Canada, pp 1365–1368. http://www.vldb.org/conf/2004/DEMP29.PDF

Statistisches Bundesamt (2011) Unfallentwicklung auf deutschen Straßen 2010. http://www.destatis.de