A Data Stream-based Evaluation Framework for Traffic Information Systems

Sandra Geisler
Information Systems
RWTH Aachen University
Aachen, Germany
geisler@dbis.rwth-aachen.de

Christoph Quix
Information Systems
RWTH Aachen University
Aachen, Germany
quix@dbis.rwth-aachen.de

Stefan Schiffer
Knowledge-based Systems Group
RWTH Aachen University
Aachen, Germany
schiffer@cs.rwth-aachen.de

ABSTRACT

Traffic information systems based on mobile, in-car sensor technology are a challenge for data management systems as a huge amount of data has to be processed in real-time. Data mining methods must be adapted to cope with these challenges in handling streaming data. Although several data stream mining methods have been proposed, an evaluation of such methods in the context of traffic applications is yet missing. In this paper, we present an evaluation framework for data stream mining for traffic applications. We apply a traffic simulation software to emulate the generation of traffic data by mobile probes. The framework is evaluated in a first case study, namely queue-end detection. We show first results of the evaluation of a data stream mining method, varying multiple parameters for the traffic simulation. The goal of our work is to identify parameter settings for which the data stream mining methods produce useful results for the traffic application at hand.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Systems and Software—*Performance evaluation (efficiency and effectiveness)*

General Terms

Design, Performance

Keywords

Data Streams, Data Mining, Traffic Information Systems

1. INTRODUCTION

Though road traffic has never been as secure as today still many people are injured or die in car crashes. On German highways, collisions with driving ahead or waiting vehicles have been the most common cause of traffic fatalities in 2009 [17]. Typical traffic scenarios which cause this type of collisions are queue-ends. Therefore, the immanent question is, how can these critical situations be detected and prevented. One method is to warn road users who approach the end of a queue and give them the chance to adapt their speed in due time. A traffic jam is not a static phenomenon – it grows or shrinks over time depending on the traffic density. Therefore, the position of the queue-end has to be updated at frequent intervals, which requires the collection and analysis of up-to-date traffic data.

Data collection in road transportation is mainly executed by two types of detection mechanisms – stationary and mobile detection. Stationary detection uses sensors which are permanently affixed to the road infrastructure, such as inductive loops, cameras or weather sensors. However, the costs for stationary detection infrastructure, its installation and maintenance are high. Hence, only a fraction of the road network can be covered by stationary detection. Mobile detection uses the vehicles themselves as "sensors" and the collected data is termed *Floating Car Data* (FCD). *Extended Floating Car Data* (XFCD) includes also in-vehicle sensors, such as rain sensors. Mobile communication technologies enable the use of vehicles to collect data and to send it to a traffic management center, as well as to exchange information between a vehicle and the road infrastructure (Car-to-Infrastructure Communication, C2IC) or between two or more vehicles (Car-To-Car Communication, C2CC).

Not only the collection, but also the processing and analysis of traffic data has to be done very fast to be useful for an efficient queue-end warning mechanism. Storing the data to disk and applying data mining methods on the stored data is not applicable for sensor data as it has to be processed in real-time. Therefore, we apply *data streams* and corresponding *Data Stream Management Systems* (DSMS) to address these challenges. A DSMS focuses on processing current data. Windows comprising a time period of the near past or a number of recently seen tuples are queried and the data is dropped after processing. Queries are defined and registered once and run frequently on the data. Also trends in the data streams can be observed, e.g., by using synopses.

Besides the data management also data analysis techniques, such as data mining, have to adapt to the specifics of data streams. Therefore, data stream mining algorithms have been proposed to account for fast and one-pass processing of data. Characteristics of data stream mining algorithms are, for example, the handling of a limited memory size, and the ability to detect concept drifts in the data. However, before data stream mining can be applied in traffic applications, the following questions need to be answered: (i) Which data stream mining methods with which parameter settings produce results in the desired quality? (ii)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
IWGS '10 San Jose, CA, USA
Copyright 2010 ACM 978-1-4503-0431-3/10/11 ...$10.00.

Which data sources and which amount of data have to be analyzed to get good results? (iii) In what traffic situations do the methods deliver satisfactory results? To answer these questions, a thorough analysis of data stream mining methods for various traffic applications is necessary.

Our contribution is an evaluation framework for traffic applications based on a data stream management system and data stream mining. We show the feasibility and efficiency of the framework by a case study in the course of the project *Cooperative Cars*[1] (CoCar). A car equipped with CoCar technology (called a CoCar in the following) sends a message over a cellular network infrastructure, when it detects a hazard, e.g., when it is braking hard, to vehicles in near vicinity or a specific spatial area, depending on the event. In our case study we use CoCar messages to determine for a road section if it contains a queue-end. The proposed framework enables us to evaluate the influence of different parameters on the detection accuracy. Especially, not only the influence of the characteristics of the road network or the traffic can be analyzed, but also special features of the data management system and data analysis can be examined. The framework uses our quality- and priority-based traffic data fusion architecture presented in [10] which is briefly explained in Section 2. Section 3 details how queue-end detection is implemented. In Section 4 results of a first evaluation are presented. Section 5 discusses related work in queue-end detection and data streams. Finally, Section 6 concludes our results so far and discusses future work.

2. EVALUATION FRAMEWORK SETUP

The proposed evaluation framework is based on a data management architecture, which we designed and implemented in the CoCar project. For simplicity, in this section we will explain only those parts of the architecture which have been used in the framework and the case study, details can be found in [10]. The goal of the architecture is to provide real-time fusion of traffic data to derive and to provide new traffic information based on fused data. Three main components constitute the architecture and are described along with their realization in the framework in the following. The overall architecture is depicted in Figure 1.

2.1 Data Sources

The architecture is designed to enable the use of multiple data sources which are integrated in the data stream management system (DSMS). The data sources are one parameter which is very interesting to evaluate in a traffic application. Which data sources are sufficient for which traffic application and what is the influence of integrating additional data sources? Special emphasis is put on data sources of mobile detection as they offer several advantages over stationary detection. One goal of our work is to find out to which extent and within which limits mobile data sources can be used for traffic applications, such as derivation of traffic data [9] or traffic state estimation.

In the CoCar project, of course one important data source are CoCars sending event-based messages in case of hazards to warn other vehicles. The second important data source are Floating Phone Data (FPD). FPD are anonymously collected positions of mobile phones. From these positions other traffic data, such as the vehicle speed, can

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

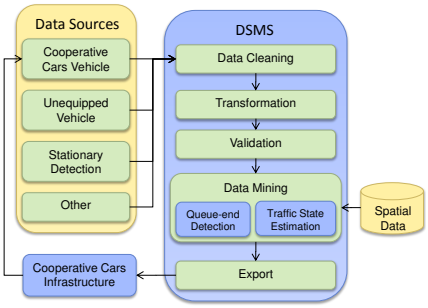


Figure 1: The overall architecture

be derived. Other data sources can be TMC-RDS messages or data from stationary detection sensors. The data can be pushed or pulled from the corresponding data source to the DSMS. For reasons of reproducibility, controllability and sufficient data creation, we use the traffic simulation software VISSIM by PTV AG[2] to emulate the creation of data from mobile and stationary sensors.

We create CoCar messages by using the VISSIM simulation for the following events: a vehicle braking very hard and a vehicle turning on the warning flashers. In our case study, we focus on the CoCar messages only as we want to analyze the usefulness of CoCar messages and parameters, such as required equipment rates, for certain traffic applications.

2.2 Data Stream Management System

In our scenario, the data is produced at high rates and hence, we need to process and analyze the data in a fast way to deliver answers timely. Therefore, we decided to use a DSMS for data processing and analysis. We use the Global Sensor Network[3] (GSN) system [1]. GSN provides a flexible, adaptable, distributable, and easy to use infrastructure. 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)[4] into the DSMS. We implemented a virtual sensor for GSN which applies a data stream mining algorithm on aggregated sensor data and outputs the results as a data stream to the DSMS. The virtual sensor is generic, i.e., the type of classifier algorithm and the corresponding configuration can be defined in a virtual sensor configuration file.

The DSMS exports the derived information and sends it as messages over the CoCar infrastructure to the connected road users, e.g., by Geocast (i.e., only into a certain geographic area). For example, the result can be a message announcing a queue-end at a certain position. The reception of messages by the vehicles might influence their driving behavior and the influence of this behavior on the corresponding traffic application can be analyzed. However, this requires a deep understanding on how drivers react to certain warning messages based on social, cognitive, and empirical studies which are out of the scope of our research. In our evaluation framework, we also export statistics for evaluation, e.g.,

[2]http://www.ptv.de
[3]http://sourceforge.net/projects/gsn/
[4]http://www.cs.waikato.ac.nz/~abifet/MOA

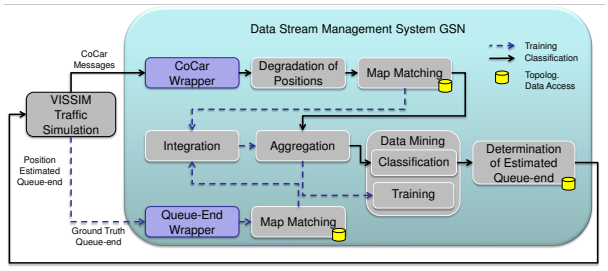


Figure 2: Overview of the queue-end scenario

mining accuracy, utilizing the export functionality of GSN.

2.3 Spatial Database

All traffic applications have spatial characteristics per definition. *Spatial databases* 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 at hand by using the spatial functionality of Microsoft SQL Server 2008[5]. We export the roads (or links) from the traffic simulation and store them as curve objects in the database. Each time we have to localize a vehicle on the network (a process termed *Map Matching*), we issue a query to the spatial database. The Map Matching and usage of spatial data in our framework is explained in detail in Section 3.

3. QUEUE-END DETECTION

To show the feasibility and effectiveness of our evaluation framework, we setup a case study scenario which aims at the detection of queue-ends. In our scenario, a road consists of two links, one for each direction. A link can be divided into sections. For each section of the links in the road network it will be determined , if it contains a queue-end or not. The scenario is aimed at investigating the influence of multiple parameters on the detection accuracy. As a source for the traffic data we use CoCar messages created by VISSIM as described in Section 2. We detail the scenario setup step by step in this section. An overview of the setup is shown in Figure 2.

3.1 Traffic Simulation

As mentioned earlier, we use a traffic simulation to create traffic data. The simulation operates on a simple road network consisting of a highway of 3km length, i.e., two links with two lanes each. The speed limit is unrestricted. The links are almost straight, with one sharp turn as depicted in Figure 3(a). One link contains a hazard (a construction site with an excavator) narrowing the street to one lane, shown in Figure 3(b). A reduced speed area encloses the construction site, restricting the maximum speed to 80km/h.

To prepare the simulation runs, the links have to be stored in the spatial database as curves to enable a Map Matching later on. As we want to identify the part of the road in which the queue-end is currently located, we divide the links of the road into sections of equal length. For each section, we will determine whether it contains a queue-end or not. The length of the section is obviously a parameter which has an influence on the accuracy of the queue-end detection. Therefore, we will investigate different section lengths in Section 4.

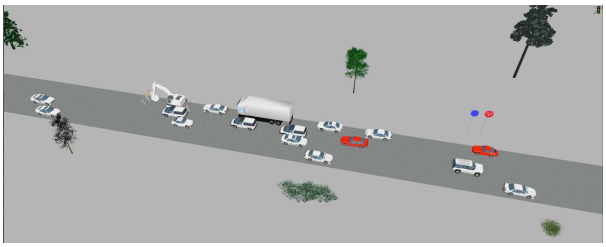
Each section is stored in the spatial database with its start point, its end point, the index of the first point and the last point of the link curve lying between start and end point of the section. This storage method ensures that the curve of a section can be easily reconstructed, but avoids redundant storage of partial curves for each section.

Since we employ learning, besides the network and the CoCar messages, we also need to determine the *correct* class for a section (does it contain a queue-end?) to train the mining algorithm, i.e., the ground truth. Hence, we added queue counters in the VISSIM road network, which are able to measure the length of a queue starting from their own position. A vehicle starts queueing when it is slower than 15 km/h and has a maximum distance of 20m to the vehicle in front. It stops queueing if it is faster than 30 km/h. The queue lengths are measured every ten seconds and if a queue is detected a corresponding message is created which contains the position of the queue-end. When the simulation is started the CoCar and queue-end messages are sent via TCP to the server hosting GSN.

3.2 Conversion, Degradation, Map Matching

The GSN system is started simultaneously with the simulation. For each message type, a separate wrapper is provided in GSN, which receives and converts the data to the GSN data format. Stream elements in GSN are defined according to a flat relational schema, i.e., comparable to datasets of a single table. The stream elements created from the CoCar messages are forwarded to a degradation virtual sensor. Each CoCar message contains, beside other information, the position, speed and acceleration of the vehicle sending the message. 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 which has been shown, e.g., in [9], and therefore, has to be considered. In case of the CoCars we can assume GPS positioning accuracy (9m error at maximum). To approximate reality as close as possible the exact positions in the messages created by VISSIM have to be degraded. To that end, we use a normal distribution to model the error. When working with real data sources this step is obsolete, of course. The accurate positions are replaced by the degraded positions in the stream elements and forwarded – all other

(a) Simulation track



(b) Hazard on the track

Figure 3: The scenario track and a hazard

[5]Other DB products provide similar functions for spatial data, thus, we are not limited to this particular product.

data remains unaltered.

The next virtual sensor matches the positions of the CoCar messages to the road network, which is termed *Map Matching*. Map Matching is the process of finding the closest link and point to the position where the road user actually is located. One very basic map matching technique is Simple Map Matching. In Simple Map Matching the road and the point with the shortest perpendicular distance to the measured point are selected. More sophisticated methods also consider the trajectory of a vehicle and the topology of the network. For a detailed survey on map matching see for example [18]. The CoCar messages do not contain any identifying information about the vehicle sending the message. Without information about the trajectory of the vehicle, we can only utilize Simple Map Matching. The Map Matching also contributes to a realistic scenario as it also introduces an error common in traffic applications.

The data in the queue-end messages represents the ground truth of the scenario. In this case, the correct position is mandatory and is not degraded. But, to determine the links and sections the queue-ends are located on, the corresponding stream elements are forwarded from the wrapper to a second map matching sensor. This sensor works in the same way as the CoCar Map Matching sensor. It introduces no error, because the positions lie exactly on the sections and are map matched precisely.

3.3 Integration and Aggregation

After the position of each message has been matched to a section, we need to prepare the data for data stream mining. The overall goal of the mining process is to determine for a set of data describing the traffic situation on the section at hand, if the section contains a queue-end. In our approach, we aggregate data of messages with positions lying on a particular section for a certain time window. This means, we calculate the following parameters from the CoCar data stream elements for one section and a time window over the last x time units:

- average speed (AVGSPEED)
- average acceleration (AVGACCEL)
- number of hard braking vehicle messages (EBL)
- number of warning light announcement messages (WLA)

In GSN there are two levels of continuous queries which are defined for assembling the output of one virtual sensor. Firstly, the required data streams currently active in the GSN system can be queried. Secondly, an overall query for joining the results of the data stream queries can be defined. This means, the aggregation and integration can be done in one virtual sensor. To complete the training dataset for the data stream mining, we have to join the aggregated data with the real queue-end stream elements, which define the true class value of each dataset. Both streams are joined over the section and the time window. The join is obsolete if no training is performed.

3.4 Data Stream Mining

In our approach, queue-end detection is casted as a binary classification task. A classifier has to decide according to the dataset at hand, whether a section contains a queue-end or not. As depicted in Figure 2, we can setup the scenario according to the two classical data mining modes: training and classification. If a classification without training is required, the obsolete virtual sensors can easily be removed. As already mentioned, we integrated the data stream mining framework MOA into GSN by encapsulating it into a virtual sensor. MOA is based on the well-known mining framework Weka[6]. In the mining virtual sensor, the incoming data stream elements are converted into MOA compatible training instances. To test the accuracy of the algorithm, we use a Test-Then-Train approach. In this approach each instance is used twice: firstly, it is used to test the accuracy and performance of the current classifier (the classifier has not seen this example yet). Secondly, it is used for the training of the classifier afterwards. This maximizes the use of the examples available and allows an accuracy analysis on a very fine-granular level [5]. The statistics of the tests comprise the number of instances used for training so far and values for true positives and negatives and false positives and negatives.

So far, we started evaluation with the implementation of a Hoeffding Tree classifier included in the MOA framework, which is "state-of-the-art for classifying high speed data streams"[5]. The Hoeffding tree splits on an attribute if the difference between the attribute's estimated information gain and the information gain of the second best attribute excels the determined Hoeffding bound. The Hoeffding bound has been originally introduced to data stream mining by Domingos and Hulten [13]. For the configuration of the algorithm a default set of options has been used, because we want to focus on the evaluation of the influence of varying traffic scenario parameters, such as penetration rate. In future research we will include the evaluation of multiple algorithms and corresponding configuration sets.

After the mining algorithm classified an instance the statistics of the classifier are updated and the classifier is trained on the instance. The corresponding stream element is enhanced with the estimated class and statistical information and passed on in the GSN system. If a section has been classified to contain a queue-end, the coordinates of the section midpoint are retrieved by querying the spatial database. Then, a corresponding CoCar message is generated containing this position and sent back to the VISSIM simulation via TCP. The estimated queue-end and the correct queue-end are visualized by blue and red traffic signs as shown in Figure 3(b).

4. EVALUATION

We demonstrate the usability and efficiency of our framework on one example traffic application. The goal of this scenario is to investigate the influence of several parameters on the accuracy of the queue-end detection using data streams, data stream mining, and CoCar messages as data source as detailed in Section 3. In this evaluation, we focus on the effect of various application-related parameters on the accuracy of the detection method. We analyzed three parameters, which seemed to be most promising in the queue-end scenario: the penetration rate of CoCars, the traffic volume, and the length of the sections. In addition, we studied one system-related parameter: the window size of the queries in the DSMS, i.e., the amount of data from the past which is taken into account by the data stream mining methods. For the queue-end detection time-based windows have been used, to select data from the last defined period of time, e.g., the last minute.

[6] `http://www.cs.waikato.ac.nz/~ml`

For the evaluation, a default configuration of these parameters has been determined. We set the penetration rate to 5%, the traffic volume to 2500 veh/h and the section length to 100 meters. The window size was chosen to be 120 seconds. To see the influence of one parameter, this parameter has been varied and all other parameters kept their default value. The following evaluations of the data mining results have been made for each run, whereby we define two classes to be identified by the classifier: sections with a queue-end, denoted as *Positives*, and sections without a queue-end, denoted as *Negatives*.

- **Overall accuracy:** Determines the ratio of instances with correctly classified classes to the number of all instances classified so far.

$$\frac{\text{True Positives} + \text{True Negatives}}{\text{Positives} + \text{Negatives}} \quad (1)$$

- **Sensitivity:** Sensitivity is the ratio of correctly identified queue-ends to the number of all real queue-ends.

$$\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (2)$$

  In this scenario, we are highly interested in the sensitivity, because it is more dangerous to leave a real queue end unrevealed as to forecast a non-existent queue-end.

- **Specificity:** Specificity is defined as the ratio of the number of correctly classified sections without queue-end to the number of all Negatives.

$$\frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}} \quad (3)$$

Each evaluation starts with a new tree, i.e., the classifier has not seen any training instances so far. To ensure comparability, all components in the framework have to work in a reproducible manner, i.e., all randomized elements have to be set to a fixed seed. For example, each traffic simulation run configured with the same parameters is identical, i.e., the raw data produced by the vehicles is always the same as well as the traffic state in each time step. To test the comparability, two identical evaluation runs with default values have been made before starting the actual tests. The runs had almost identical accuracy gradients and differed no more than 3% in value. Differences in values can be explained by minor time shifts induced by TCP transfer in GSN and differences in time between start of the simulation and start of the GSN system. That means, the time windows might not be at the exact same position in the time line as they had been in the run before and therefore, might not result in identical averages. Each simulation run had a duration of 30 minutes. The average mining times for one element lay between 10 and 20ms, while the algorithm mined up to 5833 elements per run. In the following, we describe each evaluated parameter and discuss and show the most interesting results of the experiments.

4.1 Window Size

The window size determines how much data of the past is taken into account by the data stream mining algorithm. A good window size would be very close to the average time for which the queue-end stays in one section. For example, if the queue-end is 50 seconds in section A and 50 seconds in section B, then with a window size of 100 seconds (or more),

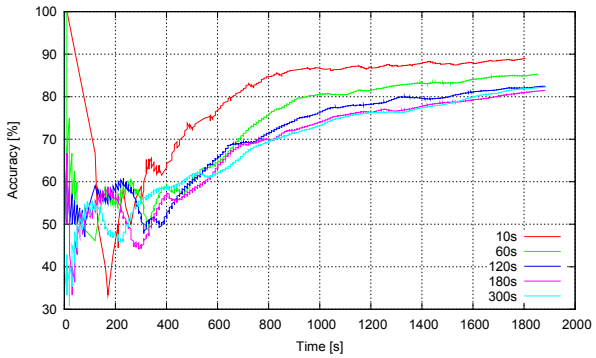


Figure 4: Accuracy for varying window sizes

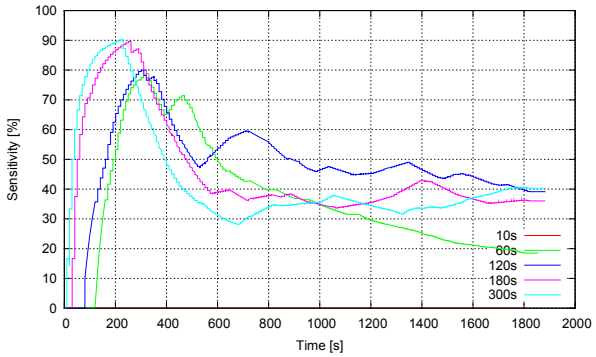


Figure 5: Sensitivity for varying window sizes

the system would not be able to make a distinction between section A and B. On the other hand, if the window size is too small, the mining algorithm will not get enough information to determine that a queue-end is in a particular section. Based on some initial observations on the time needed for a queue-end to go from one section to the next (we observed a time of about 100 to 200s for the default traffic volume), we decided to vary the window size parameter between 10 and 300s. The results for accuracy and sensitivity are shown in Figure 4 and 5.

The results for accuracy are not helpful to identify a good window size as there are much more negatives than positives, especially for the smaller window sizes. Therefore, a window size of 10s has the best accuracy although there is never a true positive (the line for a window size is not visible in Figure 5 as it is always at 0%). The results for sensitivity show that our initial estimation for a good window size was correct, as window sizes of 90s and 120s deliver most of the time the best results. The noisy start phase of about 300s should not be taken into account as the traffic jam has not been established yet and the mining algorithms could not be trained on many positives up to that point. It is interesting to see that a window size of 300s delivers in the end the best results, although the sensitivity has been much worse earlier. Furthermore, it can be observed, that sizes over 120s seem to stabilize in the end in contrast to smaller sizes, which fall all the time. We need more experiments with longer simulation times to get a conclusive result for the window size.

4.2 Traffic Volume

The traffic volume is given in vehicles per hour. It is expected that, when the traffic volume increases, more vehicles will be present in one section. This will also lead to a

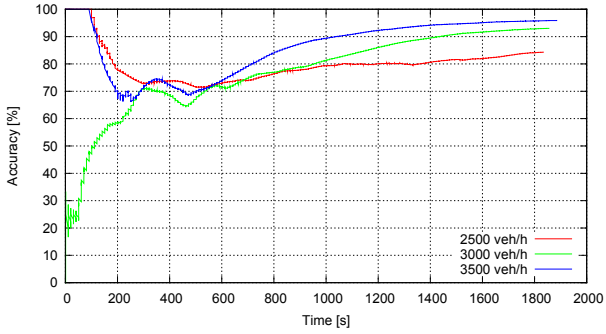


Figure 6: Accuracy for varying traffic volumes

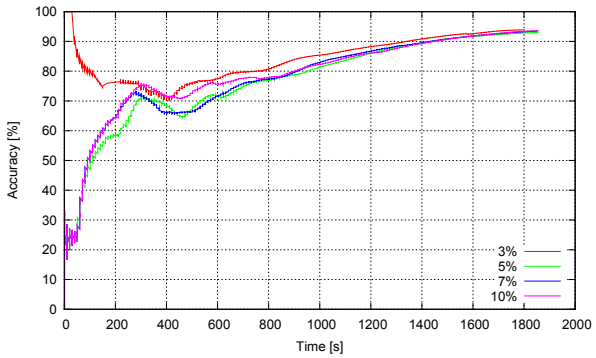


Figure 7: Accuracy for varying penetration rates

higher volume of CoCars per section and enhance the number of messages per section. Additionally, a queue will grow faster when the amount of traffic is higher, which results in a higher rate of vehicles actively braking and switching on their warning flashers. It can be assumed that a higher amount of messages will lead to a more reliable approximation of the actual traffic state in one section. Therefore, we expect the accuracy to increase when the traffic volume is rising. According to current statistics published by the German Federal Highway Research Institute[7] on German highways traffic volume averages 1500 veh/h calculated from the daily traffic volume (including the low-traffic night). However, traffic queues occur at higher volumes during the day. Therefore, we experimented with 2500, 3000 and 3500 veh/h (additionally, 1500 veh/h did not lead to any queueing in our traffic scenario). For these runs we extended our network links to 5km length, to provide enough space for the queue.

The accuracy results, depicted in Figure 6 fulfill our expectations: the accuracy rises with the increase of traffic volume. Skipping the noisy start phase, it can be observed that the accuracy is above 80% for all tested traffic volumes, even above 90% for a traffic volume of more than 3000 vehicles per hour. As expected, the highest traffic volume delivers the best results. The number of seen examples increases in each run (which is also dependent on the duration of the run). While in the 2500 veh/h run the classifier trained on 2058 instances, in the 3500 veh/h run the classifier observed 5833 examples which corroborates the assumption, that the number of messages rises with increasing vehicle volume. But the analysis of the specificity and sensitivity evaluation showed, that the biggest portion of the correctly classified instances are constituted by true negatives. The specificity reaches accuracies of 98% at maximum in the 3500 veh/h run and shows the same pattern (rising values with rising volumes), while the sensitivity only reaches about 40% in the 3000 veh/h run and no clear trend in the runs can be observed.

An analysis of the ratio of the "real" positives to negatives reveals, that with increasing traffic volume and hence, increasing queue length, obviously the portion of sections from which messages are received and are containing a queue-end in a time window decreases (from 14% in 2500 veh/h run to 4% in 3500 veh/h run), while the fraction of sections which do not contain a queue-end increases. The latter group are most likely sections with congested traffic, but do not contain the queue-end. This fact leads to reconsideration of the used attributes (Are they suited to distinguish sections with queue-ends from sections without queue-ends? Are their values close enough to real traffic situations?). In the data it is conspicuous, that emergency braking messages outweigh warning flasher messages. But braking to that extent may also occur when vehicles are in the middle of the queue. As warning flashers usually occur at the end of a queue and not in the queue, a remedy could be, to give a higher weight to the warning flasher messages in the decision process (boosting).

[7] `http://www.bast.de`

4.3 Penetration rate

One of the most common and interesting questions for traffic applications based on mobile detection is: how many vehicles must be equipped with the technology to deliver acceptable results for an application? Huber [12] identified in his analysis, that a general assumption for FCD penetration rates cannot be made, but has to be evaluated in the context of the information system, the traffic application, and the required output quality at hand. The penetration rates he identified in the analyzed studies vary between 1-5%. We took these values as a basis and made experiments with 3%, 5%, 7% and 10% penetration rate. It is expected that a higher penetration rate would lead to a higher accuracy, as it can be assumed that a higher amount of messages per section is produced. The results for this experiment were inconclusive – no clear trend could be identified in the available simulation time.

In the last section, traffic volume turned out to be a promising influence factor to enhance the accuracy. Therefore, we repeated the experiments with a traffic volume of 3000 veh/h. In Figure 7 the accuracy results are shown. Surprisingly, the experiment showed that all penetration rates converge to a value between 93% and 95%, which would lead to the assumption, that the penetration rate has no influence on the accuracy. Again, the specificity is the most influential indicator, showing the same pattern as the overall accuracy, while the sensitivity again is inconclusive: though, 10% penetration has the highest and 3% the lowest accuracy, 5% is substantially better than 7% penetration rate. More simulations would be very beneficial here to see, if this is the case also for other traffic scenarios.

4.4 Section Length

One very interesting parameter to consider is the length of the sections. The section length influences the flow rate of the system. Obviously, smaller sections lead to a higher amount of mining instances. Furthermore, with smaller sec-

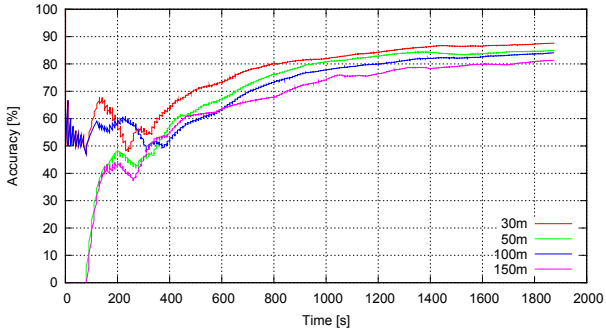


Figure 8: Accuracy for varying section lengths

tions the real queue-end can be approximated to a higher degree if classified correctly, because the mean distance between the real queue-end and the middle of the section (the forecasted queue-end) is smaller. Due to the lack of empirical data, we selected section lengths in a way, that results in a tolerable mean error of distance to the real queue-end (at most half of the section length). We experimented with 30, 50, 100 and 150 meters. It is expected that a maximum in accuracy is identified for a certain section length. Too small sections produce a too small amount of messages which are not sufficient to determine whether there is a queue-end or not. Too long sections are expected to be harmful to accuracy, because they can contain too many messages, which do not indicate the correct class. We present the results in Figure 8. Surprisingly, it can be observed, that a shorter section length seems to be more beneficial to accuracy. But, the analysis of the specificity and sensitivity shows again, that negatives have the highest impact on the result as the number of positives in the overall training instances only reaches 12%.

In conclusion, the first results of the experiments show trends of influential parameters and reveal promising accuracy values also for smaller penetration rates. However, the results are not sufficient to draw final conclusions as we need to evaluate different traffic scenarios and fine tune the data stream mining algorithm. Especially, the low sensitivity rates and inconclusive results need further investigation. However, it has been shown, that the framework can be used for the evaluation of traffic applications in a very flexible way. It can easily be extended, to investigate other parameters or applications.

5. RELATED WORK

Data stream management systems and data stream mining is a young research area and found its way into traffic applications in recent years. For example, the product MineFleet [14] integrates data stream mining into vehicle embedded systems for fleet monitoring to analyze vehicle health, emissions or driver behavior. Another approach detecting driver behavior is presented by Horovitz et al. [11], which use a combined approach of unsupervised data stream clustering and fuzzy logic to detect drunken driver behavior. Liu et al. [16] propose a distributed traffic stream mining system for determination of congestion level using Frequent Episode Mining. On a central server frequent patterns are determined based on historical data and are then distributed to stationary detectors, which use the pattern to classify data from the sensors. The benchmark Linear Road benchmark for DSMS in [3] allows for performance measurements (throughput and response time) and comparison of multiple DMS. The benchmark is based on a simulated traffic scenario which calculates tolls for vehicles driving into areas with congestion and accidents. The traffic scenario is a means to the end of producing data for system stress tests, but does not aim at analyzing the utilized traffic application. Hence, complex tasks, such as mining, are not analyzed, because it was not given prominence to the realistic simulation of a traffic scenario. Furthermore, they do not take into account possible effects of the communication system or feedback to the vehicles. Also global players in data management work on solutions for traffic applications based on data stream management. Recently, IBM proposed a real-time traffic information management system based on its streaming platform Infosphere Streams [4]. They used GPS data from taxis and trucks of the city of Stockholm to calculate travel times and shortest paths between city parts to demonstrate the effectiveness of their system. Microsoft integrates handling of spatial data with data stream capabilities implemented in StreamInsight [2].

For mobile detection of traffic data, there have been several studies, which aim at rating the accuracy of the created traffic information. Especially, in the field of anonymously collected Floating Phone Data the interest is high. These studies can be divided mainly into field studies and simulation studies. Intensive simulation studies have been carried out by Fontaine et al. [7, 8]. 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 a data stream management system, such as window size, sliding step, and of data stream mining, such as concept shift.

The investigation of queue-end detection algorithms is another important line of work. Huber [12] studied the opportunities in traffic information collection using XFCD. He analyzed, which in-vehicle information, e.g., collected over CAN bus or sensors, can be used to detect certain local traffic events. One example he investigated is the queue-end detection using a deceleration parameter, an indicator for low speed level, right turn signals, the road type, and warning flashers. The event determination is implemented using fuzzification and a rule base. To reach 90% probability that a vehicle arrives at an incident in a one minute interval a penetration rate of 6.9% is required already. In contrast, our system would be able to announce a queue-end in a much lower time interval (10s) and is therefore more suitable for local hazard warnings. Furthermore, the system does not use any learning. Fuzzification and rules are manually defined. The use of fuzzy rules would be an interesting extension to our method. In [6] Chan et al. present a system for real-time queue tracking based on the average speed detected on road sections by identifying three traffic zones and analysing the arrangement of these zones. Though, the authors base their work on stationary detection and have to rely on larger sections (minimum of 500m between loop detectors on a comprehensively equipped highway), the idea of identifying the parts of the queue seems appealing and

could be integrated in our system to enhance the quality of the detection mechanism. More current work on queue-end detection uses methods from the field of artificial intelligence. In [15] Khan presents a simulation study also based on VISSIM using stationary sensors as data source. They analyze the data 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.

6. CONCLUSION

In this paper, we presented a framework for simulation-based evaluation of traffic applications. The framework consists of a traffic simulation, a data stream management system, data stream mining algorithms, and a spatial database system. It enables us to evaluate the influence of parameters of road networks and traffic scenarios as well as the impact of characteristics of the information system and the data mining algorithms. In a case study on queue-end detection, we illustrated that our approach yields feasible means to investigate effects of these parameters. The results of the evaluation carried out show, that trends in accuracy and specificity based on varying parameter configurations can be observed already with small sets of training instances. Results for sensitivity, an indicator for the detection accuracy for queue-ends, were inconclusive in the tests, most probably due to the small ratio of positives in comparison to negatives in the training instances and also the high number of map matching errors (almost 47% for default values). While traffic volume seems to be a strong factor influencing data mining accuracy, the penetration rate showed no crucial effect. The section length had a high impact on the accuracy, caused by the small number of messages and the even higher difference in the number of positives opposed to negatives, which has been reflected in the results for the sensitivity. As the flexibility of the framework enables the analysis of the effect of a variety of parameters in the data management system and the data stream mining, further evaluations are the next step in our work plan. Additionally, we will investigate means to enhance the accuracy and sensitivity of queue-end detection, by enhancing Map Matching accuracy, by boosting certain input attributes, or by including data of adjacent sections. in the decision process. Also, taking into account other data attributes and data sources for the mining algorithms is easily possible in our data fusion architecture. In conclusion, it has been shown that the framework is a very good instrument to study traffic applications and the corresponding information system.

7. ACKNOWLEDGMENTS

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.

8. REFERENCES

[1] K. Aberer, M. Hauswirth, and A. Salehi. A middleware for fast and flexible sensor network deployment. In *Proc. VLDB'06*, pages 1199–1202, Seoul, 2006.

[2] M. H. Ali, B. Chandramouli, B. S. Raman, and E. Katibah. Spatio-Temporal Stream Processing in Microsoft StreamInsight. *IEEE Data Eng. Bull.*, 33(2):69–74, 2010.

[3] A. Arasu, M. Cherniack, E. Galvez, D. Maier, A. S. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts. Linear road: a stream data management benchmark. In *Proc. of VLDB'04*, 2004.

[4] A. Biem, E. Bouillet, H. Feng, A. Ranganathan, A. Riabov, O. Verscheure, H. Koutsopoulos, and C. Moran. IBM InfoSphere Streams for Scalable, Real-Time, Intelligent Transportation Services. In *Proc. of SIGMOD'10*, 2010.

[5] A. Bifet and R. Kirkby. *Data Stream Mining - A Practical Approach*. University of Waikato, 2009.

[6] T. N. Chan, C. Lee, and B. Hellinga. Real-time Identification and Tracking of Traffic Queues Based on Average Link Speed. In *Transportation Research Board*, 2003.

[7] M. D. Fontaine and B. L. Smith. Probe-based Traffic Monitoring Systems with Wireless Location Technology. *Transportation Research Record*, 1925:3–11, 2005.

[8] M. D. Fontaine and B. L. Smith. Investigation of the Performance of Wireless Location Technology-Based Traffic Monitoring Systems. *Journal of Transportation Engineering*, 133:157–165, 2007.

[9] S. Geisler, Y. Chen, C. Quix, and G. Gehlen. Accuracy Assessment for Traffic Information Derived from Floating Phone Data. In *Proc. ITS'10*, 2010.

[10] S. Geisler, C. Quix, G. G. Gehlen, and G. Jodlauk. A Quality- and Priority-based Traffic Information Fusion Architecture. In *Proc. ITS'09*, 2009.

[11] O. Horovitz, S. Krishnaswamy, and M. M. Gaber. A fuzzy approach for interpretation of ubiquitous data stream clustering and its application to road safety. *Intelligent Data Analysis*, 11:89–108, 2007.

[12] W. Huber. *Vehicle-generated Data for Aquiring Traffic Information*. PhD thesis, TU München, 2001.

[13] G. Hulten, L. Spencer, and P. Domingos. Mining time changing data streams. In *Proc. KDD'00*, 2000.

[14] H. Kargupta, K. Sarkar, and M. Gilligan. MineFleet: An Overview of a Widely Adopted Distributed Vehicle Performance Data Mining System. In *Proc. KDD'10*, 2010.

[15] A. Khan. Intelligent Infrastructure-based Queue-end Warning System for Avoiding Rear Impacts. *IET Intelligent Transport Systems*, 1:138–143, 2007.

[16] Y. Liu, A. Choudhary, J. Zhou, and A. Khokhar. A Scalable Distributed Stream Mining System for Highway TrafficData. In *PKDD 2006, LNCS 4213*.

[17] Statistisches Bundesamt. Verkehr - Verkehrsunfälle. `http://www-ec.destatis.de`, July 2010.

[18] C. E. White, D. Bernstein, and A. L. Kornhauser. Some Map Matching Algorithms for Personal Navigation Assistants. *Transportation Research Part C: Emerging Technologies*, 8:91–108, 2000.