Learning Driving Behavior by Timed Syntactic Pattern Recognition

Sicco Verwer
Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001
Leuven, Belgium
sicco.verwer@cs.kuleuven.be

Mathijs de Weerdt and **Cees Witteveen**
Delft University of Technology
Mekelweg 4, 2628 CD
Delft, the Netherlands
m.m.deweerdt@tudelft.nl, c.witteveen@tudelft.nl

Abstract

We advocate the use of an explicit time representation in syntactic pattern recognition because it can result in more succinct models and easier learning problems. We apply this approach to the real-world problem of learning models for the driving behavior of truck drivers. We discretize the values of onboard sensors into simple events. Instead of the common syntactic pattern recognition approach of sampling the signal values at a fixed rate, we model the time constraints using timed models. We learn these models using the RTI+ algorithm from grammatical inference, and show how to use computational mechanics and a form of semi-supervised classification to construct a real-time automaton classifier for driving behavior. Promising results are shown using this new approach.

1 Introduction

Processes often contain constraints on the timing of events (such as deadlines). When learning a model for such a process using for instance syntactic pattern recognition [Fu, 1974] or hidden Markov models (HMMs) [Rabiner, 1989], these time constraints are modeled *implicitly* by sampling the signal values at a fixed rate, resulting in a blowup of both models and events. Using *explicit* state duration models such as hidden semi-Markov models (HSMMs) [Yu, 2010] avoids this blowup but does not model the time constraints directly. Recently, the idea of learning *real-time automata* has been proposed [Verwer *et al.*, 2010]. Since probabilistic automata can model only a strict subset of the distributions of HMMs, these real-time automata are less powerful than traditional HSMMs. However, the advantage of such models is that they do allow their *structure* including time constraints to be learned efficiently [Verwer *et al.*, 2009]. To the best of our knowledge, learning real-time automata has never actually been applied to real data. By learning these models for driving behavior, we show in this paper that this results in useful models for use in practical applications. In addition, we develop the necessary preprocessing and postprocessing methods. Our approach can be considered a first attempt at *timed syntactic pattern recognition*.

The data at our disposal consists of onboard sensor measurements that have been collected from truck round-trips. By applying a simple discretization method, we obtain sequences of timed events. The behavior that is displayed in these sequences is unknown. From this data, we want to learn a model that we can use to monitor the driving behavior in new data, i.e., to use it as a classifier. Our approach is to first learn a timed model from the unlabeled sequences using the RTI+ algorithm [Verwer *et al.*, 2010]. Subsequently, we create a small number of labeled examples by displaying some typical behaviors during a test lap. We use these to label some states of the learned model. The labels of these states are used to classify new data. This is a novel method to perform classification when many unlabeled and few labeled examples are available; the setting of semi-supervised classification [Zhu and Goldberg, 2009]. This setting occurs in many practical applications because it is often too time-consuming to assign labels to the observed examples.

In addition to there being mostly unlabeled data, there is an additional restrictive property in our application domain: the process under observation is continuous, i.e., it never stops producing events. Consequently, we need to be able to learn a model from a single but very long unlabeled event sequence. We use the theory of computational mechanics [Shalizi and Crutchfield, 2001] to show why sampling random subsequences from this single example provides sufficient information to learn a useful model, the so-called causal structure of the process. This model can be learned by RTI+ by providing the random subsequences as input.

We apply our approach to a test case of detecting bad driving behavior when accelerating after a complete stop. After learning a timed model and labeling some of the states using 23 labeled accelerations, we constructed a simple decision rule based on the number of times the labeled states are visited. We also implemented a simple decision rule based on expert knowledge. These two decision rules agree in 79% of all acceleration cases in the original unlabeled data. This surprisingly accurate result serves as a proof-of-concept for timed syntactic pattern recognition.

This paper is structured as follows. We first describe the background of our application, its goals, and why we choose timed syntactic pattern recognition in order to achieve these goals in Section 2. Afterwards, in Section 3.1, we briefly introduce the RTI+ algorithm, and show how to use it in order

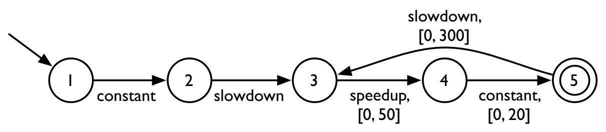

Figure 1: The harmonica driving behavior modeled as a DRTA. The intervals on the state transitions are bounds on the amount of time (in tenths of a second) the behavior can remain in the current state directly before firing the transition.

to learn a classifier for driving behavior. Our results using this approach on truck acceleration are discussed in Section 4. We end this paper with a general discussion and some ideas for future research in Section 5.

2 Motivation

We are interested in methods that learn models for the behavior of a system (process). A well-known model for characterizing systems is the *deterministic finite state automaton* (DFA, see, e.g. [Sipser, 1997]). An advantage of an automaton model is that it is easy to interpret. In contrast, many other well-known models that are commonly used to learn the behavior of a system are quite difficult to interpret, for example neural networks or support vector machines, see, e.g., [Bishop, 2006]. Since a DFA is a language model, its learning (or identification) problem has been well studied in the grammatical inference field [de la Higuera, 2005]. Hence, we would like to apply an established method in order to learn a DFA from observations of the system. However, when observing a system there often is more information than just the sequence of symbols: the *time* at which these symbols occur is also available. A straightforward way to make use of this timed information is to *sample* the timed data. For instance, a symbol that occurs 3 seconds after the previous event can be modeled as 3 consecutive event symbols. This technique is commonly used when applying syntactic pattern recognition [Fu, 1974] or hidden Markov models [Rabiner, 1989] to signal processing. However, since this technique models time *implicitly*, i.e., using additional states, it can lead to an exponential blowup (requiring n states instead of $\log(n)$ bits) of both the input data and the resulting model.

We use of a different method that uses the time information directly in order to produce a timed model. A common model for real-time systems is the *timed automaton* (TA) [Alur and Dill, 1994]. Unfortunately, there exists no learning algorithm for TAs in general. We therefore use a restricted type of TA known as a deterministic real-time-automata (DRTAs) [Dima, 2001] for which an efficient learning algorithm exists [Verwer *et al.*, 2010]. A DRTA contains constraints on the time between two consecutive events. Using these constraints, the language of a DRTA does not only depend on the types of events occurring, but also on their time values relative to previous event occurrences. We clarify this using an example from our application domain of constructing models for truck driver behavior:

Example 1 *Figure 1 shows an example DRTA that models a specific driving behavior known as 'harmonica driving'. This often occurs when a truck is driving at a somewhat higher speed than the vehicle directly in front of it (no time constraint). The driver slows down a bit (no time constraint), waits until there is enough distance between him and the vehicle in front (takes at most 50 seconds), and then speeds up again and closes in on the vehicle (for at most 20 seconds). This whole process often repeats itself a couple of times (within 5 minutes) before the driver finally adjusts the speed of the truck to match the vehicle in front of him. The result of this whole process is unnecessary fuel consumption.*

The example clearly shows the importance of time information for modeling driving behavior. Time constraints model the dependence on this time information *explicitly*, i.e., using numbers. Because this uses a binary representation of time, it can result in exponentially more compact models than an implicit representation. Consequently, also the time, space, and data required to learn DRTAs can be exponentially smaller than the time, space, and data required to learn DFAs [Verwer *et al.*, 2009]. This efficiency argument is our main reason for using DRTA models. In the next section, we briefly describe the algorithm we use to learn these DRTA models, and show how to use it in the practical setting of learning driving behavior.

3 Learning DRTAs

The RTI+ algorithm for learning DRTAs [Verwer *et al.*, 2010] is a kind of model selection technique. It requires as input a set S_+ of timed strings (time-stamped event sequences) that have been generated by a process (system), and returns a DRTA model $\mathcal{M}$ that is consistent with S_+. Intuitively, a model $\mathcal{M}$ is *consistent* if all timed strings S that reach the same state in $\mathcal{M}$ do not have significantly different futures in S_+. In other words, $\mathcal{M}$ is a DRTA that satisfies a Markov property. This property is tested using likelihood ratio tests on S_+. The goal of RTI+ is to find the smallest consistent model $\mathcal{M}^*$. Unfortunately, finding $\mathcal{M}^*$ is very difficult (NP-hard). RTI+ is a greedy method that finds good solutions efficiently.

We apply RTI+ to the problem of automatically constructing model that can detect different driving behaviors from sensor measurements. As can be seen in Example 1, DRTAs provide succinct and insightful models for such behavior. We now describe the challenges that need to be overcome in order to learn such models based on data from onboard sensors.

3.1 Data discretization

We first need to *discretize* the sensor measurements into basic events (symbols). The data we obtained from the trucks were 15 complete round trips. One round trip typically takes two full days. These trips were driven on different dates, with different weather conditions, and with different drivers. We focus our attention on three sensor values that were recorded during these trips: *speed*, *engine rotations per minute* (RPM), and *fuel usage*. These measurements were recorded at a rate of one per second. Figure 2 shows one hour of these sensor measurements.

Ideally, we would like to transform these measurements into driving actions/events such as: speed-ups, slow-downs,

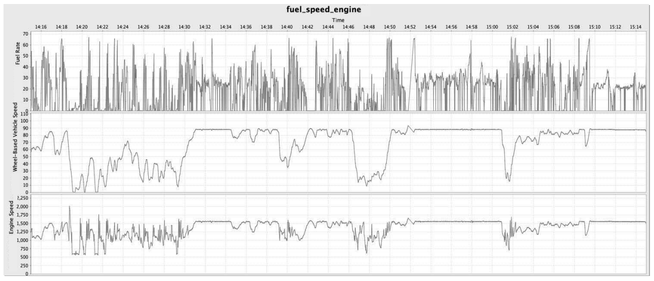

Figure 2: The values of three sensors installed on trucks. The first graph shows the fuel rate in liters per hour. The second shows the vehicle speed in kilometers per hour. The third shows the engine speed in rotations per minute.

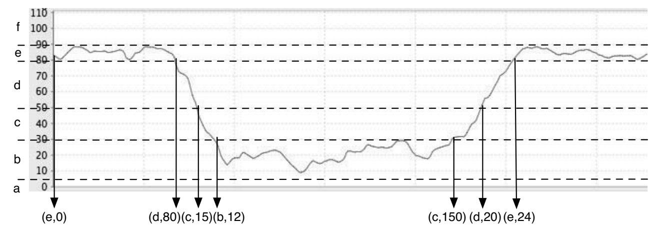

Figure 3: The discretization routine used by our algorithm. Whenever the sensor value exceeds a predetermined bound, plus or minus a small additional region (in this case an additional 2 km/h), an event is generated corresponding to the region the value has entered. The time value of the event is the time that has elapsed since the previous event.

a turn left/right, a bump in the road, etc. However, since finding patterns in the sensor values that are indicative of these kinds of events is far from trivial (see, e.g., [Roddick and Spiliopoulou, 2002]), we discretize the data in a much simpler way. We divide the range of a sensor value into different regions, and whenever the value enters a region, we treat it as the occurrence of an event associated with that region. The time value of the occurrence is the time that has elapsed since the previous event occurrence. We use expert knowledge to create an intuitive division of the sensor values into such regions. The discretization process for the speed sensor is depicted in Figure 3. The additional region of $2km/h$ ensures that the event occurrence does not change constantly when the sensor value is near a region boundary.

Using the discretization procedure, we obtain from every round-trip of a truck, and for every sensor, a very long sequence of timed symbols. RTI+ requires a data set S_+ of preferably many not too long timed strings. We transform the discretized data to such a set S_+ using theory from computational mechanics.

3.2 Selecting subsequences

Computational mechanics [Shalizi and Crutchfield, 2001] models a *process* as an bi-infinite sequence of random variables (observations). The goal of computational mechanics is to predict the (infinite) future of a process based only on a finite observation sequence, i.e., a single long sequence of events. Learning a predictive model requires a set of positive example strings S_+. This set is created by selecting (random or all) subsequences of the observed sequence. In order to generalize (learn) over such arbitrary subsequences, we have to make the assumption that the process is stationary. This assumption seems restrictive, but it is necessary since the subsequences in S_+ are allowed to start and stop at arbitrary points in time. Consequently, important information for a non-stationary process (regarding event occurrences before the starting event) is unavailable. Although driving behavior is not a typical example of a stationary process, this assumption does allow us to learn useful models.

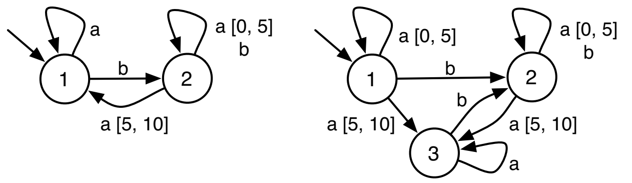

Figure 4: A deterministic real-time automaton $\mathcal{A}$ (left) and its causal states $\mathcal{A}'$ (right). States 1 and 2 of the original model correspond respectively to states 3 and 2 of the causal model.

The subsequences in S_+ can be used to learn what are called the *causal states* in computational mechanics. Essentially, the causal states form a DRTA that represents behavior (a probability distribution over timed strings) identical to a non-deterministic version of the original DRTA. This non-deterministic version has an identical structure as the original DRTA, but has the complete set of states as possible start states. We illustrate this concept for the problem of learning a DRTA using an example.

Example 2 *In Figure 4, a DRTA is depicted along with its causal states. The causal states also form a DRTA. The causal start state corresponds to all (or any) of the states of the original DRTA. In other words, we do not know whether the original DRTA is in state* 1 *or in state* 2. *The occurrences of events can be used to determine the state of the original DRTA. For example, the occurrence of a* b *event ensures that the next state is state* 2, *since all transitions with* b *as label have state* 2 *as their target. Similarly, the occurrence of an* a *event with a time value greater than* 5 *ensures that the next state is state* 1. *When an* a *event occurs with a time value less than* 4, *we do not know whether* 1 *or* 2 *is the next state. Thus, in this case, the target state in the causal DRTA is the state that corresponds to both state* 1 *and* 2, *i.e., the start state.*

Since the causal states are the deterministic equivalent of a non-deterministic automaton, it is possible that the blow-up in the number of states is exponential. This is the price we have to pay for learning the causal states instead of the original automaton. We have no choice, however, because the original DRTA can only be learned from timed strings that are guaranteed to start in the start state.

Combining all the data we have available, we obtain for the speed sensor about $50,000$ symbol occurrences in to-

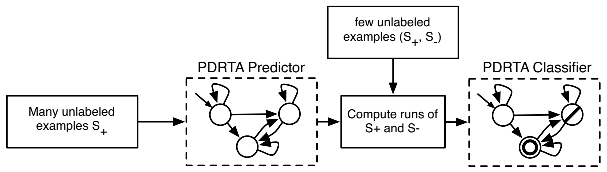
few unlabeled examples (S+, S-)
PDRTA Predictor
PDRTA Classifier
Many unlabeled examples S+
Compute runs of S+ and S-

Figure 5: Constructing a classifier based on a PDRTA. We use the runs of a few labeled examples to assign labels to the states of the PDRTA. The result is a classifier.

tal. For the engine and fuel sensor we obtain about twice this amount. From this data, we took $10,000$ random subsequences of length 20. We now explain how we use this data in order to construct a classifier for driving behavior.

3.3 Learning a DRTA classifier

The result of running RTI+ on an unlabeled data set is a *probabilistic DRTA* (PDRTA). This automaton does not contain any final states and hence can initially not be used to classify new data. However, the PDRTA model does describe the behavior that is displayed in the unlabeled data set. If learned correctly, two timed strings only reach the same states if their possible futures are identically distributed. Intuitively, these two timed strings display the same behavior, and hence their labels should also be the same. Consequently, we can use the label of a single labeled string to label a state of the PDRTA. This process is depicted in Figure 5.

Since RTI+ learns the causal states instead of the actual states of the PDRTA, the states that are visited by good behavior b and bad behavior b' can overlap. After some events however, different behaviors will display different future distributions and hence the visited states will be different. Let Q_b and $Q_{b'}$ denote the sets of states that are reached by the timed strings that display behavior b and b' respectively. We use these sets of states to build our classifier as follows:

- assign the label b to the states in $Q_b \setminus Q_{b'}$, i.e., the states reached by b but not by b';
- assign the label b' to the states in $Q_{b'} \setminus Q_b$, i.e., the states reached by b' but not by b.

When a new timed string reaches a state labeled with b, we conclude that it is displaying behavior b, and not b'. In addition to classifying new timed strings, the labels assigned to states can also be used to analyze the old unlabeled data set, for instance, in order to determine how often good or bad driving occurred in the past.

In this way, RTI+ learns one classifier for every sensor. However, since we model the causal states of the process, it is unclear what part of a new timed string to use in order to determine the classification. We could use the whole timed string, i.e., simply compute the complete run and determine the label at every index. But this only works if the learned model is completely correct. When we make some small error during learning, the run of the new timed string can diverge from its actual run and from that point on many classifications will be incorrect. Thus, the run of a shorter sequence will be more correct since it is based on more data (larger future distributions) during learning, and the run of a longer sequence will be less correct but better at distinguishing the different types of behavior. We decided to use all sequences of length 1 to 10 and combine their classifications using a simple decision rule based on the amount of reached good and bad states:

$$\mathsf{value}(i,j) = \begin{cases} 1 & \text{if } \mathsf{label}(s,i,j) = good \\ -1 & \text{if } \mathsf{label}(s,i,j) = bad \\ 0 & \text{otherwise} \end{cases}$$

$$\mathsf{score}(i) = \mathsf{score}(i-1) + \sum_{0 \le j \le 10} \mathsf{value}(i,j)$$

where $\mathsf{label}(s,i,j)$ is a function that returns the label of the subsequence from index $i-j$ to index i of a new sequence s, and $\mathsf{score}(0) = 0$. By summing these score values over all sensors we obtain a classifier that can be used to determine the type of driving behavior at every index i of a new sequence of driving events.

	size	AIC	AIC single state model
speed	$2,305$	$871,550$	$1,472,840$
fuel	960	$842,536$	$1,102,270$
engine	$1,009$	$697,190$	$1,076,750$

Figure 6: The sizes and AIC scores of the PDRTAs (lower is better) learned by our algorithm for every sensor.

4 Results

We applied the RTI+ algorithm to the discretized data and learned one PDRTA model for each of the three sensors. The performance of each of these models is measured using their sizes and their Akaike information criterion (AIC) [Akaike, 1974] values. These measure a trade-off between model size and likelihood (lower is better). The sizes and AIC scores of each of these models and a trivial single state model are displayed in Figure 6. From the AIC scores, we can conclude that the models perform a lot better than the trivial single state model. Other than that, the AIC scores by themselves do not say much about the quality of the learned PDRTA models. However, the sizes of the learned models allow us to draw some conclusions regarding this quality.

Unfortunately, these sizes are quite large. From results using RTI+ described in [Verwer *et al.*, 2010], we know that it is capable of correctly inferring a PDRTA with about size 40 from a data set of size $2,000$. From the truck data, we obtained PDRTAs that are much bigger. Hence, it is very unlikely that the learned PDRTAs are completely correct. Although we did learn these PDRTAs using $10,000$ instead of $2,000$ examples, this increase in data does not explain the large size of the learned models. The first 50 states of the PDRTAs (in distance from the start state) however are learned using a lot of data. Because of cycles in the transitions of the PDRTAs, most of these states are reached by around $5,000$ timed strings. Since $2,000$ timed strings is sufficient to correctly learn the start state of a size 40 PDRTA, these first states are very likely to be correct. We therefore use these first 50 states of the PDRTAs as classifiers.

accelerating too fast	normal acceleration
$(a,42)(b,80)(c,12)(d,10)(c,7)$	$(a,10)(b,14)(c,21)(b,42)$
$(a,13)(b,65)(c,6)(d,8)$	$(a,11)(b,34)(c,13)(b,18)$
$(a,6)(b,39)(c,10)(d,10)$	$(a,8)(b,10)(c,14)(b,45)$
$(a,7)(b,15)(c,8)(d,7)$	$(a,25)(b,5)(c,18)(b,27)$
$(a,7)(b,2)(c,7)(d,7)$	$(a,11)(b,6)(c,20)(b,14)$
$(c,22)(b,4)(c,12)(d,7)(c,8)$	$(a,12)(b,18)(c,15)(b,20)$
$(a,7)(b,11)(c,6)(d,7)(c,16)$	$(a,8)(b,35)(c,24)(b,20)$
$(a,8)(b,29)(c,7)(d,8)(c,8)$	$(a,12)(b,32)(c,18)(d,13)$
$(a,11)(b,28)(c,7)(b,12)$	$(a,7)(b,62)(c,21)(b,28)$
$(a,10)(b,320)(c,12)(d,7)(c,34)$	$(a,12)(b,8)(c,17)$
$(a,10)(b,8)(c,8)(d,8)(c,31)$	$(a,10)(b,27)(c,19)(d,16)$
$(a,8)(b,24)(c,9)(d,8)(c,12)$	

Figure 7: Events obtained from the speed sensor during a test-lap with both too quick and normal accelerations.

4.1 The labeled examples

For the detection of truck driver behavior, we tested our approach on a simple type of behavior that results in unnecessary fuel usage: accelerating too quickly. This behavior serves as an example of how to construct a classifier out of the learned PDRTAs using only a few labeled examples. Using the same techniques, the learned PDRTAs can be used to classify many other interesting driving behaviors.

In order to classify accelerations, we require some labeled examples. The labeled examples were obtained by driving a short test lap with many traffic lights. After a full stop, the truck driver accelerated too fast about half of the time, and the other half he accelerated normally. We labeled the timed events that occurred during $1^1/_2$ minute from the start of the acceleration, Figure 7 shows these events for the speed sensor. We used the same time span in order to obtain examples for the fuel and engine sensors.

All but one of the timed strings in Figure 7 start with an a symbol. This means that the truck came to a full stop (or at least a speed less than 5 km/hour). In one timed string the initial symbol is a c symbol. In this case, the driver actually did not slow down completely, but accelerated too fast from driving around 20 km/hour (the next event is a b event). In every timed string, the second symbol is a b symbol, which indicates a speed-up of the truck. The time until this symbol occurs is the number of seconds the truck remained stationary at the traffic light. The third symbol is a c event, indicating a further speed-up. It should be possible to use the time value of this event in order to classify the two different acceleration behaviors: a small time value indicates a fast acceleration, a large value indicates a slow acceleration. This can be clearly seen in the examples. The next event of the examples is either a b or a d event, depending on whether the speed of the truck exceeded 50 km/hour or not. A small number combined with a d event indicates that the driver is accelerating too quickly. A b event indicates that the driver is slowing down again, for instance in order to stop for the next traffic light.

Having collected these labeled examples, all we need to do is to run the learned PDRTAs on these examples, and discover which states are reached when accelerating too quickly, and which when accelerating normally. Figure 8 shows these

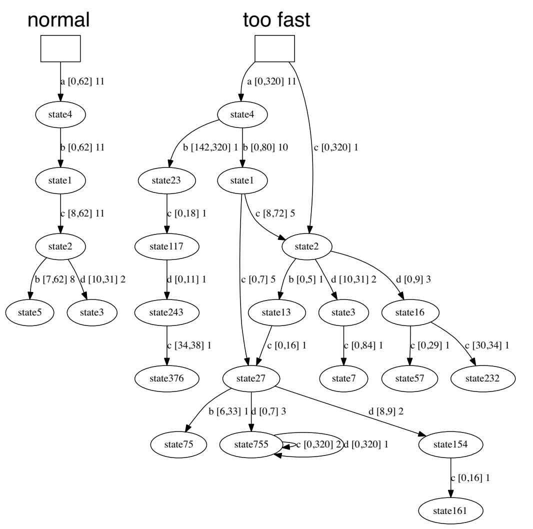

Figure 8: Parts of the learned PDRTA for the speed sensor that are reached by the timed strings from Figure 7. A labels, time constraint, and the number of reaching strings are shown for each transition.

states and the traversed transitions for the speed sensor examples. We used these PDRTA parts to identify some states that are only reached when the driver accelerates too quickly and some when he accelerates normally. In Figure 8, state 5 is reached by 8 of the 11 normal examples, and none of the 12 fast examples. An example of a state that is only reached by the fast examples is state 27; 6 of the fast examples reach this state. Note that the transition to this state has c as symbol and $[0, 7]$ as clock guard. Thus, it is only reached by timed strings that contain a c event within 7 seconds after the first b. A c event with more than 7 seconds goes to state 2, which is also the state that is reached by all the normal acceleration examples. Thus, the learned PDRTA model for the speed sensor does distinguish between the two behaviors of accelerating too fast and accelerating normally, exactly in the way we expected based on the timed strings of Figure 7.

This shows that the models learned by RTI+ can be used to distinguish different types of behavior, even if we beforehand do not know what behavior to distinguish (the PDRTAs are learned from unlabeled data). In this way, timed syntactic pattern recognition can be used to give more insight into the different kinds of behavior of a system/process.

For all PDRTAs, we identified and labeled all states that were reached by at least 4 example strings in one case, and by none in the other case. Thus, for the speed sensor state 27 and 16 are labeled in the too-fast speed PDRTA, and state 5 is labeled in the normal speed PDRTA. We did the same for the fuel sensor and engine sensor PDRTAs.

4.2 Classifier performance

We tested the obtained classifiers on the historical data that we have available. This is the unlabeled data that we initially used to learn the PDRTAs. From this data, we extracted all acceleration instances and used this data to first determine the correlation of our classifiers with the truck speedup. The truck speedup is the speed increase between two consecutive measurements of the speed sensor. In our tests, the correlation between these speedups and our classifiers turns out to be high only for the fuel sensor. This is surprising since only 4 of the approximately 200 states in this classifier are labeled.

We would like to determine the quality of the fuel classifier. Unfortunately, we do not exactly know which of the unlabeled accelerations are too fast, and which are not. Instead of determining the exact values, we therefore implemented a simple method that is currently used by domain experts to determine a driver's profile to approximate these values:

- Over 30 seconds, count the number of times that the acceleration was normal n (less than $0.8 m/s^2$) and the number of times it was too quick q (greater than $1.2 m/s^2$).
- If the n is less or equal to 25, and q is greater or equal to 1, then label the acceleration as too quick; label it as normal otherwise.

The two numbers n and q give a good overview of a driver's profile. The fuel classifier labels an acceleration as too quick if its score is greater or equal to 1. We compare the labels assigned by the fuel classifier and this simple decision rule:

		decision rule	
		quick	normal
classifier	quick	920	61
	normal	262	289

The labels given by these two rules match in 79% of all cases. This is rather high considering we only used 23 example strings to label only 4 states of the fuel PDRTA.

5 Discussion

We applied timed syntactic pattern recognition techniques to the problem of detecting driver behavior. Although our techniques are still preliminary, the result that labeling only 4 states in a learned PDRTA is sufficient to obtain an adequate classifier clearly shows the high potential of our techniques. It would be interesting to investigate whether other driving patterns can also be distinguished using these PDRTAs.

Unfortunately, since the learned PDRTA models are quite large, it is difficult to discover different driving behaviors by examining the models. We aim to use tools from process mining [van der Aalst and Weijters, 2005] for this purpose. Process mining tools contain many routines and visualizations that help to make large process models insightful.

Interesting directions for future work are the construction of more sophisticated discretization routines, better decision rules and more specialized automaton models. In addition, our techniques are also useful for learning other models. It would for instance be interesting to see whether our semi-supervised method also achieves good results with learning algorithms for non-timed automata such as the state-merging algorithms for DFAs (see, e.g., [de la Higuera, 2005]). Furthermore, it would be interesting to see whether our techniques can also be applied to methods for learning types of hidden semi-Markov models (see, e.g., [Yu, 2010]) where the transition probabilities depend on the state durations by means of time constraints.

References

[Akaike, 1974] Hirotsugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[Alur and Dill, 1994] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[Bishop, 2006] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[de la Higuera, 2005] Colin de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 38(9):1332–1348, 2005.

[Dima, 2001] Catalin Dima. Real-time automata. *Journal of Automata, Languages and Combinatorics*, 6(1):2–23, 2001.

[Fu, 1974] King Sun Fu. *Syntactic Methods in Pattern Recognition*. Academic Press, 1974.

[Rabiner, 1989] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, 1989.

[Roddick and Spiliopoulou, 2002] John F. Roddick and Myra Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):750–767, 2002.

[Shalizi and Crutchfield, 2001] Cosma Rohilla Shalizi and James P. Crutchfield. Computational mechanics: pattern and prediction, structure and simplicity. *Journal of statistical physics*, 104(3-4):817–879, 2001.

[Sipser, 1997] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing, 1997.

[van der Aalst and Weijters, 2005] Wil M.P. van der Aalst and A.J.M.M. (Ton) Weijters. Process mining. In *Process-Aware Information Systems: Bridging People and Software through Process Technology*, pages 235–255. Wiley & Sons, 2005.

[Verwer *et al.*, 2009] Sicco Verwer, Mathijs de Weerdt, and Cees Witteveen. One-clock deterministic timed automata are efficiently identifiable in the limit. In *LATA*, volume 5457 of *LNCS*, pages 740–751. Springer, 2009.

[Verwer *et al.*, 2010] Sicco Verwer, Mathijs de Weerdt, and Cees Witteveen. A likelihood-ratio test for identifying probabilistic deterministic real-time automata from positive data. In *ICGI*, volume 6339 of *LNCS*, pages 203–216. Springer, 2010.

[Yu, 2010] Shun-Zheng Yu. Hidden semi-markov models. *Artificial Intelligence*, 174(2):215 – 243, 2010.

[Zhu and Goldberg, 2009] Xiaojin Zhu and Andrew B. Goldberg. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130, 2009.