Clustering Based On Driving Style Using Hot Paths

By Henrik Ullerichs, Lynge Poulsgaard, and Philip Sørensen

Summary of Master Thesis

This thesis explores the use of GPS data in evaluating and grouping drivers based on their driving behavior. The motivation for this project is the increasing amount of GPS data available, more and more users utilize GPS enabled devices and the data generated from these devices could be used in many aspects, such as driver behavior analysis. This is a very interesting subject to users such as insurance companies who are interested in more accurately assessing their customers' risk based on their actual behavior, rather than statistical analysis, which is common today.

Driving behavior can be assessed by looking at the data associated with the GPS data, such as data from accelerometers, compasses, etc. but also from data derived directly from the GPS data itself. This paper focuses on the later of the two, based on the 217 GB ITS dataset generated in 2012-2014 by 458 drivers located in the North Jutland region of Denmark.

The thesis describes a framework able to evaluate drivers based on any dataset containing GPS data. To fairly evaluate drivers we introduce the term Hot Paths, describing heavily traversed paths in the road network, which is ideal to use as a common denominator when evaluating driver behavior, as they reveal paths containing large amounts of data.

The Hot Paths reveal traffic flow in the road network and visualizes good candidates to use when comparing drivers, as using data gathered on a Hot Path can ensure that the drivers have had similar conditions when driving, e.g. the driven route, time of day, and so on, removing one of the many variables when comparing driver behavior.

The thesis explores three different methods for generating Hot Paths, a modified Apriori algorithm based on the driven trips of drivers, a graph based algorithm based on the topology of the road network, and an algorithm using Strict Path Queries to generate Hot Paths, using precomputed hashes to quickly find routes in the network. The thesis explains the reasoning for considering these algorithms along with the technical aspects of each algorithm.

The thesis evaluates the different implementations for finding Hot Paths and finds that the modified Apriori algorithm is the most efficient in terms of performance, achieving a runtime orders of magnitudes faster than the two other implementations when evaluated on the ITS dataset. These tests also shows that the framework is able to handle large amounts of data in an efficient manner.

The thesis then moves on to explain how the framework allows the user to cluster the data on driving behavior for a selected Hot Path to ensure fairness in the clustering and evaluation. The thesis explores different variations of clustering methods using k-means and DBSCAN with and without the use of the t-Distributed Stochastic Neighbor Embedding dimensionality reduction method.

The thesis also explains the use of various clustering scoring methods as the data lacks a ground truth for the problem of defining driving behavior. As such the data is evaluated on how good the resulting clusters are based on four different scoring methods, namely Dunn, Davies-Bouldin, Silhouette and Calinski-Harabasz.

Using these methods the thesis then evaluates the results of the various combinations of clustering techniques on the dataset using an exhaustive Grid Search trying to maximize the clustering scores to get the best cluster representation. The thesis finds that it is possible to distinguish different driving behavior by looking at the resulting clusters and even more so that the clustering can be done in near-realtime making the framework usable in realtime applications.
The thesis concludes that the main contributions is an efficient framework for generating Hot Paths from GPS data and evaluating driver behavior using clustering methods based on these Hot Paths to ensure a fair evaluation.

Clustering Based On Driving Style Using Hot Paths

Henrik Ullerichs
Aalborg University
huller15@student.aau.dk

Lynge Poulsgaard
Aalborg University
lkpo12@student.aau.dk

Philip Sørensen
Aalborg University
ppsa12@student.aau.dk

ABSTRACT

Driving behavior have shown to have an impact on proneness to vehicular accidents and fuel economy making it very interesting for insurance companies and fleet owners. By analyzing GPS data from drivers it is possible to evaluate and group drivers based on their driver behavior, measured in terms of acceleration, jerk, lateral acceleration, and wobble. To fairly group and evaluate driving behavior we introduce the term Hot Path, which is heavily traversed paths in a road network. Using Hot Paths, as a common denominator, we utilize k-means and DBSCAN along with the dimensionality reduction technique t-SNE to cluster driving behavior based on the observed data for each Hot Path. This paper presents a framework for generating Hot Paths for a large dataset of GPS trajectories using a novel variant of the Apriori algorithm. The framework is afterwards able to create meaningful clusters based on the observed data for the Hot Paths giving a user the tools to more easily evaluate and compare drivers using map matched GPS data. Through experimentation and evaluation using various cluster scoring methods we show that the framework is able to efficiently and effectively handle large datasets and find meaningful clusters in the data, e.g. representing calm and aggressive driving behavior.

Keywords

Insurance premium, fleet owners, clustering drivers, Hot Path, GPS data

1. INTRODUCTION

Driving style have shown to have an impact on the likelihood of vehicular accidents, as shown in [12]. Previously insurance companies, fleet owners, and similar, had to rely on the number of accidents and general personal information, such as age and sex, to assess the risk of the driver based on statistics. With the increased usage of telematics, which is the use of satellite navigation, computers, and telecommunication in vehicles, it is now possible to gather huge amounts of data on each driver's behavior. This makes it possible for both insurance companies and fleet owners to monitor and evaluate exactly how their drivers perform [33].

Insurance companies have an interest in accurately calculating the car insurance premium for its customers to make money, but also to have competitive pricing in order to retain customers. There are already a few insurance companies, such as ingenie [25], insurethebox [16] and Co-op Insurance [15], who use telematics to monitor their customers and reward them for good driving behavior. Fleet owners can also save money by observing and encouraging their drivers to exhibit driving behavior that promote lower risk, maintenance cost and fuel consumption. Although drivers will have their own individual driving style, other drivers might have similar driving behavior. To find out if this is the case we will examine possible clusters of drivers based on similar driving style, e.g. hard braking. These clusters can then be used as a basis for evaluation and comparison of drivers when selecting premiums in an insurance company, or for giving targeted training to fleet drivers. An example of a clustering can be seen in Figure 1 where there are 3 distinct clusters, each of them consisting of drivers with common driving styles.

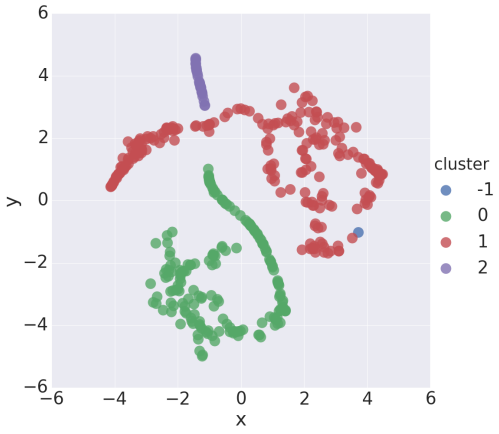


Figure 1: Different drivers being clustered into 3 distinct clusters, and a single outlier (−1).

Most existing methods for evaluating and comparing driver behavior only looks at each drivers general data, and does not consider specific contexts like the driven roads, time, or weather [8]. Not considering these circumstances can lead to unfair comparisons of drivers, as these conditions can have an effect on the drivers behavior. While there are existing methods that do consider the specific context of the data, they choose the specific roads based on existing domain knowledge, rather than the collected data [29].

In order to compare drivers efficiently and fairly we introduce the notion of a Hot Path. That is a heavily traversed path in a road network. The idea is that the data from Hot Paths can therefore be used as a basis for fairly comparing drivers. Examples of Hot Paths can be seen in Figure 2,

where Hot Paths from the city of Aalborg in Denmark are highlighted. Unsurprisingly many main roads, such as motorways, are highlighted, with only minor Hot Paths on the access roads. A major difference to just using the main roads for comparison between drivers is that using Hot Paths we guarantee that each trip have traversed the entirety of the Hot Path, while this is not the case when using the entirety of a main roads directly. Hot Paths are also usable for evaluating a road network as the flow of traffic and potential bottlenecks of the network becomes apparent. Furthermore, we append Hot Paths with a context, e.g. Monday morning, making it a more customizable tool for analyzing the driver behavior and flow in the road network.

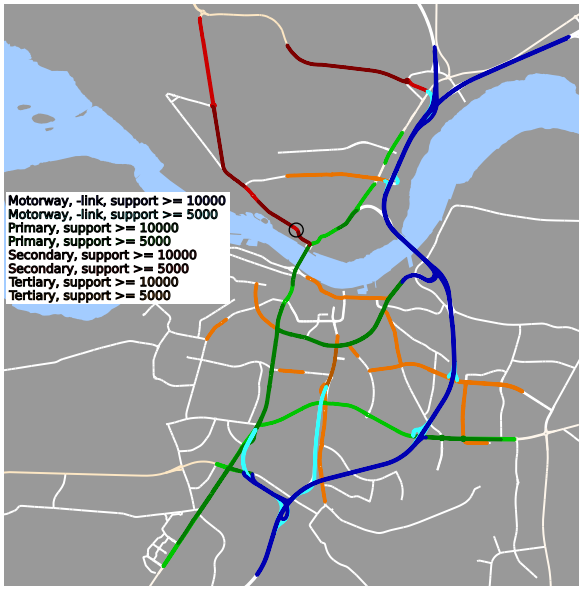


Figure 2: Hot Paths with more than 5.000 in support in Aalborg, Denmark.

Analyzing GPS data can yield information about acceleration, jerk, lateral acceleration (speed through corners), wobbling (drifting back and forth), speeding, and possibly congestions.

The goal of this paper is to propose an efficient data mining framework for comparing and grouping drivers based on their driving behavior obtained through GPS data using Hot Paths as a common denominator. Drivers can then be compared in a fair way as they are all driving on selected Hot Paths. The main contributions of this paper are:

– An efficient framework for finding Hot Paths using a novel variant of the Apriori algorithm.

– Driving style evaluation using clustering, based on data from Hot Paths.

The remaining part of the paper is structures as follows: The related work is explored in Section 2. Section 3 provides an overview of the framework. The term Hot Path is formally defined in Section 4, where we also present different algorithms for finding Hot Paths. In Section 5 the data and parameters used when determining driver behavior is defined. In Section 6 we discuss different methods for clustering. In Section 7 we present the different experiments and results to evaluate our solution. Subsequently we present and conclude on our findings in Section 8. Lastly in Section 9 we outline the possible extensions of our work.

2. RELATED WORKS

To cluster drivers on their behavior fairly using Hot Paths, we need to look into driving behavior and Hot Paths. In each of these categories we will introduce existing work.

Driving Behavior

Castignani et al. [5] looked into driver behavior profiling using data from smartphones, including GPS data and motion sensor data. In creating driver profiling they utilize metrics such as jerk and yaw. They show that jerk and yaw are efficient at identifying calm and aggressive driver behavior, making it interesting to consider these metrics for this paper as well. Compared to us, they do not consider lateral acceleration and use a coarse grouping of measures, where we use the actual values when clustering.

Murphey et al. [29] takes a similar approach by analyzing the jerk of drivers, which confirms that this could be an interesting feature to look at. Their dataset is, however, created through simulated driving with Powertrain System Analysis Toolkit, whereas our dataset consists of real driving data.

Constantinescu et al. [8] models driving style through statistical analysis, including mean and standard deviation values, for driving parameters such as speed, acceleration, braking, and mechanical work. They then apply Hierarchical Cluster Analysis and Principal Component Analysis to find meaningful groupings of drivers. This paper shows that grouping drivers based on statistical analysis of the driving behavior is interesting, which we will also explore in this paper. They do not consider any behavior related to steering, where we consider both lateral acceleration and wobbling.

In some of our previous work [19] we have worked with extracting driver preferences from driving behavior quite similar to this paper. The main difference here is that in the previous paper we were interested in the drivers preference with regards to new routes based on how the driver have driven previously, whereas this paper focuses on comparing and evaluating drivers based on how they have driven.

Clustering Drivers

Kalsoom et al. [18] have very similar goals to ours, they want to cluster drivers based on obtained GPS data, in either a slow, normal or fast driving style category. They evaluate both k-means and hierarchical clustering where their results show that k-means yields the best results. From this we can conclude that hierarchical clustering might not be a suitable method for our project, whereas k-means might be interesting. In addition to k-means we also use DBSCAN and dimensionality reduction.

Wang et al. [34] presents a solution to cluster drivers into two categories, aggressive and moderate. They have measured vehicle speed and throttle opening for each driver. The solution is based on a k-means SVM model using k-means to identify the two categories and building an SVM model to predict future occurrences. Their results are promising which indicate that k-means is effective in identifying clusters, and that SVM is able to accurately predict the classification of new entries. A large difference is that they utilize data from a driving simulator, as opposed to data obtained from physically driving on a road network.

Breuß et al. [3] explore clusterings based on GPS data with regards to energy consumption. They utilize a jerk based feature value along with a hierarchical clustering algorithm to cluster on energy consumption of vehicles based on driving data. They find that the jerk based clustering yields quite good results, making jerk an interesting feature to investigate for clustering.

Hot Path / Most Frequent Path

Luo et al. [26] developed a framework for querying time period-based most frequent paths from big trajectory data. Essentially this is very similar to our notion of a Hot Path, however, their solution is point-to-point where as our implementation finds all such Hot Paths for the entirety of an area, e.g. a city. Furthermore, finding Hot Paths are not the goal of this project, we utilize them as the foundation of our comparison of drivers.

Krogh et al. [22] presents Strict Path Queries, a novel way of querying large amounts of trajectory data. They propose a method making it possible to get all trajectories for each distinct path between two arcs, while only having to read data for the first and last arc of the path. This increases performance greatly when having to find all trajectories traveling on a specific path between two nodes in the road network. This kind of querying and optimization techniques might be interesting for the work with Hot Paths as well, however there is one large difference, as we are interested in analyzing all Hot Paths for the entire road network, where Strict Path Queries are point-to-point.

Li et al. [24] develops an algorithm, FlowScan, for finding so-called "hot routes" in a road network with moving objects, quite similar to how we find Hot Paths based on trajectories in a road network. Li et al. does not work on an actual dataset, but instead rely on generated data for San Francisco. To generate interesting paths they add neighbourhoods to the map and generate trips in-between these. Hot routes differ from Hot Paths as hot routes allow some slack in the route, e.g. two people driving on a slightly different path could be included in the same hot route. Hot Paths on the other hand requires the trajectories to follow the same path to create a proper base for comparing driving behavior.

3. FRAMEWORK OVERVIEW

This section describes the overall components of the framework we develop before diving into the specific parts. The complete overview is seen in Figure 3, and reveals that our framework closely resembles the Extract, Transform, Load (ETL) process.

The framework can be used with most map matched GPS datasets and can be used to compare and evaluate individual driving styles. Each of the major processes in our framework are briefly explained in the following, whereafter the detailed description will come in the following sections.

Preprocessing

Within the preprocessing step we take raw GPS data logs and "enrich" the data. The derived data we have used, such as lateral acceleration, is defined in Section 5.1.1.

Furthermore, this is also the step where we calculate the Hot Paths for driver comparison in a later step. Hot Paths are defined in Section 4.

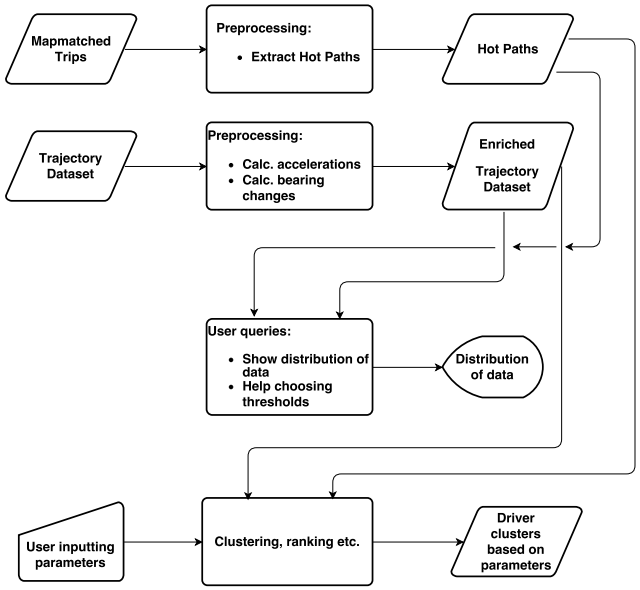


Figure 3: A complete overview of our framework. Slanted boxes are datasets, square boxes with slanted roof are user input, and square boxes are processes. The "distribution of data" is a visual presentation.

User Queries

This process generates data and visualization of the data to help the user choose suitable parameters for both Hot Path and the subsequent clustering. However, this is outside the scope of this project and will not be explored in detail. OpenJUMP can be used as a more technical way to visualize the data and help the user choose suitable parameters.

Clustering

In this process the clustering algorithms runs on the enriched trajectory dataset, and the resulting clusters are scored accordingly to their quality. Details about the clustering algorithms and the scoring method we implement are described in Section 6.

Furthermore, it should be noted that adding both new clustering techniques and rating algorithms is relatively straightforward.

4. HOT PATHS

An interesting problem for users like insurance companies is to find a fair way to asses the risk profiles of drivers. This can be done by analyzing how different drivers perform when driving on the same route. To facilitate this, we first need to find which routes are used heavily by multiple drivers. We call these highly used routes; Hot Paths. In the following section we define the various notions we use when calculating Hot Paths, including the structure of our GPS data, road network, and Hot Paths along with the associated features, such as support and context.

4.1 Definitions

A *road network* is a directed graph $G = (N, A)$ containing nodes and arcs. A *node* $n \in N$ corresponds to either a shared point between two or more road segments or the end of a road segment. Each *arc* $a = (s, e) \in A$ represents a directed

road segment from the node s to the node e. The start and end node of an arc is denoted $a.s$ and $a.e$ respectively.

A *trajectory* $Tr = \langle g_1, g_2, \ldots, g_n \rangle$ is a sequence of GPS points, ordered by time, representing a single trip. Each GPS point g is defined as a tuple:

$$g = (v, timestamp, x, y, speed, acc, jerk, \theta, weather)$$

where v is the vehicle id, timestamp includes both date and time, x and y represents a GPS coordinate, speed is the vehicular speed in km/h, acc is the acceleration in m/s^2, jerk is the change in acceleration per second m/s^3, θ is the bearing of the vehicle measured in degrees and weather is the associated weather, as measured by the nearest weather station, including temperature, rainfall, wind, etc.

The trajectories in the ITS dataset are map matched to the road network using the Viterbi algorithm proposed by Wei et al. [35]. This method uses Markov models to find the most likely path in the road network based on a given trajectory. A map matching is mapping a subsequence of a trajectory to the most likely arc along with an id denoting the given trip so that all map matchings with a given trip id can be collected to get the map matched path for the entire trajectory. A map matched trajectory is defined as:

$$\begin{aligned} M &= \langle m_1, m_2, \ldots, m_n \rangle \\ m \in M, a \in A, m &= \left(v, t, a, Tr'\right) \\ Tr' &= \langle g_i, g_{i+1}, \ldots, g_k \rangle \end{aligned} \tag{1}$$

where v is the vehicle id and t is the corresponding trip id. Furthermore the subtrajectory, Tr', of a map matching cannot overlap with any other subtrajectory used in other map matchings of the map matched trajectory.

We define a function $match(a)$ that returns all map matchings for a given arc, that is:

$$match(a, \mathbb{M}) = \left\{ m \in \mathbb{M} \;\middle|\; m.a = a \right\} \tag{2}$$

Where $\mathbb{M}$ is an optional parameter, if no parameter is supplied the set of all map matchings for the entire dataset is used.

A *path* $P = \langle a_1, a_2, \cdots, a_n \rangle, n \geq 1$ is a sequence of distinct adjacent arcs such that $\forall i, 1 \leq i < n,\ a_i.e = a_{i+1}.s$ and $\forall i \in [1; n], \neg\exists j$ s.t. $i \neq j \wedge a_i = a_j$.

A *sub path* $P' = \langle s_1, s_2, \ldots, s_k \rangle$ of $P = \langle a_1, a_2, \ldots, a_n \rangle$ exists when $|P'| < |P|$ and P' is completely contained within P, that is $\exists i$ s.t. $\langle a_i, a_{i+1}, \ldots, a_{i+k-1} \rangle = P'$.

If P' is a sub path of P then P is also called the *super path* of P'.

We use the term *support*, borrowed from Apriori [1], as the number of observations on a path. In this paper, we use *trip support* and *vehicle support* to refer to the number of observed trips and the number of different vehicles respectively.

The set of trips passing an arc a, $T(a)$, and the set of vehicles passing an arc a, $V(a)$ is defined as:

$$\begin{aligned} T(a) &= \{m_1.t, m_2.t, \ldots, m_k.t\}, m_i \in match(a) \\ V(a) &= \{m_1.v, m_2.v, \ldots, m_k.v\}, m_i \in match(a) \end{aligned} \tag{3}$$

The trip support of a path $P = \langle a_1, a_2, \ldots, a_n \rangle$ is defined as $S_{trip}(P) = \left| \bigcap_{i=1}^{n} (T(a_i)) \right|$ and the vehicle support of P is given by $S_{vehicle}(P) = \left| \bigcap_{i=1}^{n} (V(a_i)) \right|$.

The *context*, C, is some user defined filter that works on the data associated with all trips on a path. In the ITS dataset filters can use date, time, and weather. Specific data can also be derived from the associated data, e.g. defining seasons and peak hour times to only consider trips driven in winter and during peak hours. The context is user defined, and can be absent. The context is defined as a set of context functions, $f(P)$, that takes a path, P, and returns either true or false:

$$\begin{aligned} C &= \langle c_1, c_2, \ldots, c_n \rangle, n \geq 0 \\ c_i &= f(m) \\ f(m) &= \begin{cases} true \\ false \end{cases} \end{aligned} \tag{4}$$

A context function could be looking at the season, weekday, date, weather, etc. of each path. Such a function looking at the season could be:

$$\begin{aligned} &isSeason(season', m) = \\ &\quad \left\{ g_1 \in m.Tr' \;\middle|\; season(g_1.timestamp) = season' \right\} \end{aligned} \tag{5}$$

To be usable context functions only have one input, namely the path. As such the *isSeason* function has to be instantiated with a season, for example *isWinter*:

$$isWinter(m) = isSeason(winter, m) \tag{6}$$

In SQL these functions translate to a *where* clause with *and* in between the expression of all context functions.

We say that a path, P, belongs to a context, C, if the following holds.

$$\forall a \in P, \forall m \in match(a), \forall c \in C, c(m) = true \tag{7}$$

A specific context associated with a path can be denoted by $P.C$, the path P must belong to any such context.

When limiting the context of Hot Paths we actually limit the map matched data for that given Hot Path. As such we define a function *limitToContext*, seen in Equation 8, to generate the set of map matchings limited to a given context.

$$\begin{aligned} &limitToContext(\mathbb{M}, C) = \\ &\quad \left\{ m \in \mathbb{M} \;\middle|\; \forall c \in C, c(m) = true \right\} \end{aligned} \tag{8}$$

4.1.1 Hot Path

A *Hot Path*, H, is a path having a user-defined minimum trip support and vehicle support defined as:

$$\begin{aligned} H = &\ \langle a_1, a_2, \cdots, a_n \rangle, \\ \text{where } &S_{trip}(H) \geq minTripSupport \\ &\wedge\ S_{vehicle}(H) \geq minVehicleSupport \end{aligned} \tag{9}$$

The order of arcs in the Hot Path is the order they were used when driving, i.e. if all drivers take one path in the morning, and takes the reversed path on the way back, that will constitute two different Hot Paths.

4.2 Finding Hot Paths

In this section we introduce three methods for finding Hot Paths and argue for the advantages and disadvantages of each one. We present two similar graph based algorithms

and a third using the Strict Path Query method developed by Krogh et al. [22].

We implement our algorithms using SQL and PL/pgSQL with a few C functions needed where the functions supplied by PostgreSQL are insufficient, or where we gain a performance improvement compared to PL/pgSQL.

Apriori

To find Hot Paths we have developed a novel variant of the Apriori algorithm [1] that utilizes the driven sequences of arcs. One feature of the Apriori algorithm is, that it will generate Hot Path candidates of all consecutive arc combinations, having sufficient support. As we need this data in order to choose appropriate support levels for the ITS dataset, this algorithm is a suitable choice for our purposes. The original Apriori algorithm may generate huge candidate sets, because all items may be combined with any other item, but because we only combine with the next arc in the direction driven, the size of candidate sets actually shrink as the Hot Paths grow, because candidates are removed due to lack of support or no more consecutive arcs are available. Compared to the original Apriori algorithm, we modify the candidate generation step. Our algorithm, which is listed in Algorithm 1, has a list of trips as input, where each trip has a trip id and an array of traversed arcs: $\{tripId, arcs[]\}$. As input and during the execution of the algorithm, the arcs are in the order they were traversed. The first step is generation of $1-itemset$, where we generate one item for each passage of each arc. When we calculate the candidates during each iteration, we differ from the original Apriori algorithm. Because the arcs are not independent of the trajectory, only itemsets respecting the traversal order are usable as candidates. When transitioning from the $k-itemset$ to the $(k+1)-itemset$, we add the next arc in the order they were traversed. For example, if the trajectory visits the arcs <a_1,a_2,a_3> in that order, the $1-itemset$ has the items {<a_1>, <a_2>, <a_3>}, the $2-itemset$ has the items {<a_1,a_2>, <a_2,a_3>} and the $3-itemset$ has the item {<a_1,a_2,a_3>}.

The user can limit which arcs to include, either by specifying an input table containing only the wanted arcs, or by passing one or more arc filters to the Hot Path function. Available filters enable the user to limit by road type, e.g. exclude unpaved roads, by supplying a geometry (bounding box), e.g. only arcs inside the Aalborg/Nørresundby city area or by ignoring the start and end of each trip. The user can further limit the amount of saved data using the parameters minimum and maximum Hot Path Size (number of traversed arcs), and minimum Hot Path length (Hot Path length in meters).

If the trips are not filtered by context before the Hot Paths are generated, the Apriori algorithm finds Hot paths for all chosen context combinations. The end-user can then query as needed afterwards.

Graph Based

As an alternative to the Apriori algorithm, we extract Hot Paths using the graph representation of a road network. The algorithm listed in Algorithm 2 uses the data annotated with each arc to extract Hot Paths in a breadth first manner. Any arc where trip and vehicle support are above the minimum will itself be a Hot Path. From these it is possible to expand the Hot Paths by adding the following arc. This new

```
Input: Trips: {(tripId, context, vehicle, <arcs>)}
Input: Maximum Hot Path size: maxHPSize
Input: Minimum Hot Path size: minHPSize
Input: Minimum Hot Path length: minHPLength
Input: Output table name: outTable
Input: Minimum trip support: minTripSupport
Input: Minimum vehicle support: minVehicleSupport
Input: Columns for context: context
Input: Arc filters: arcFilter
Data: Arcs with sufficient trip support: CT0
Data: Candidate tables: CT1, CT2

/* Extract arcs with sufficient support
   (1-itemsets)                                    */
foreach trip do
  for i in 1..length(arcs) do
    if arcFilter(arcs_i) then
      if S_trip(arc) >= minTripSupport then
        insert into CT0 (tripId, arcKey, path, arcNo,
          nextArcNo, pathLength, context, vehicle)
        values(tripId, arcs_i, <arcs_i>, i, i + 1,
          arcLength, context, vehicle)
      end
    end
  end
end

/* Generate k-itemsets                             */
sourceTab := CT0
destTab := CT2
nextDestTab := CT1
repeat
  truncate table destTab
  insert into destTab
    select p1.tripid,
           p2.nextArcNo,
           p1.context, p1.vehicle,
           p1.path || p2.arcKey as path,
           p1.pathLength+p2.pathLength as pathLength
      from sourceTab p1, CT0 p2
     where p1.tripId = p2.tripId
       and p1.nextArcNo = p2.arcNo
       and S_trip(path) >= minTripSupport

  if Hot Path size >= minHPSize then
    insert into outTable
      select tripid,
             S_trip(path) as tripSupport,
             S_vehicle(path) as vehicleSupport,
             context, vehicle, path
        from destTab
       where vehicleSupport >= minVehicleSupport
         and pathLength >= minHPLength
  end

  /* swap source and dest                          */
  sourceTab := destTab
  destTab := nextDestTab
  nextDestTab := sourceTab
until Hot Path size = maxHPSize or no more
  candidates
```

Algorithm 1: Novel Apriori algorithm for finding Hot Paths.

Hot Path is then added to the set of Hot Paths, if its trip and vehicle support are above the minimum. This iterative process is continued until no arcs can be added to any Hot Path.

If a specific context is considered, the map matched data can be limited to that context using the *limitToContext* function.

Input: Graph of the road network: G(N,A)
Input: Minimum trip support: $minTripSupport$
Input: Minimum vehicle support: $minVehicleSupport$
Input: context: C
Data: Hot Paths: $HP := \{\emptyset\}$

```
/* Finding first Hot Paths                                  */
foreach a ∈ A do
    P := ⟨a⟩
    if S_trip(P) ≥ minTripSupport
    ∧S_vehicle(P) ≥ minVehicleSupport then
        HP := HP ∪ {P}
    end
end

/* Expanded Hot Paths                                       */
pathLength := 1
repeat
    foreach H := ⟨a_1, ..., a_n⟩ ∈ HP do
        if |H| = pathLength then
            foreach a ∈ A do
                if a_n.e = a.s then
                    P := H ∪ a
                    if S_trip(P) ≥ minTripSupport
                     ∧S_vehicle(P) ≥ minVehicleSupport
                     then
                        HP := HP ∪ {P}
                    end
                end
            end
        end
    end
    pathLength++
until no Hot Path is added to HP
```

Algorithm 2: Graph Based algorithm for finding Hot Paths.

Strict Path Query

A third method of extracting Hot Paths is to use the Strict Path Query (SPQ) by Krogh et al. [22]. We choose this approach because the ITS dataset is already optimized for this query type. The SPQ is highly optimized towards answering queries about a specific segment pair, but this query only tells the number of trips grouped by unique paths, but not which segments were used between the start and the end segment. Because of this we modify the query to return the actual segments used. While modifying the query, we take care to preserve as much of its advantage over a plain lookup of the start and end segments.

A main reason for SPQ's high performance is that the algorithm only has to read data for the start and end segments. However, as we need information from the intermediate segments, some performance loss is unavoidable. The performance can be somewhat preserved as we only need to lookup the intermediate segments for one trip for each unique path to get the traversed arcs.

By preserving the advantage resulting from only doing lookup of intermediate segments once for each unique path, we lose the ability to calculate the number of different vehicles, because in order to so, we need to read data for all trips to count the number of different vehicles. This also means that context functions are not supported by this implementation as well, as this also requires information from each trip individually. The primary purpose of this implementation is to compare its performance to the other implementations. As such we do not put in unnecessary work in this implementation, as a full implementation of the SPQ based algorithm would most likely lower the performance of the implementation further.

It is possible to cut the number of possible combinations in half as SPQ returns the result for both directions of a pair of source and destination. As such we only need to query for one direction for each pair.

Removing Redundancy

Common for the above methods is that they find all Hot Paths with the given minimum support. By definition all sub paths of a Hot Path are also Hot Paths. In case only the longest Hot Paths are needed, the sub paths, and thereby the redundancy, can be removed after the Apriori algorithm has been completed, but often these have higher support, so some information is lost if they are simply removed. We let the user decide which sub paths to preserve by supplying a *difference* parameter, which defines the minimal relative difference between super and sub path needed to preserve the sub path (Equation 10). We preserve a Hot Path if the relative difference in support is higher than the minimum for all of its preserved super paths.

$$
\begin{aligned}
D_{trip} &\geq \frac{S_{trip}(P_{Sub}) - S_{trip}(P_{Super})}{S_{trip}(P_{Super})} \\
D_{vehicle} &\geq \frac{S_{vehicle}(P_{Sub}) - S_{vehicle}(P_{Super}))}{S_{vehicle}(P_{Super})}
\end{aligned}
\tag{10}
$$

where

P_{Sub} is the sub path,
P_{Super} is the super path,
D_{trip} is the relative difference in trip support,
$D_{vehicle}$ is the relative difference in vehicle support

Consider the Figure 4, showing three paths. If the minimum support is 30.000, and the blue (starting at a_1) and red (starting at a_2) each has a support of 15.000 while the green (starting at a_3) has a support of 3.000, we have four Hot Paths: (33000, $\langle a_5 \rangle$), (33000, $\langle a_6 \rangle$), (33000, $\langle a_5, a_6 \rangle$) and (30000, $\langle a_4, a_5, a_6 \rangle$). The Hot Path (33000, $\langle a_5, a_6 \rangle$) will be preserved if the difference parameter is 10 % or less but Hot Paths (33000, $\langle a_5 \rangle$) and (33000, $\langle a_6 \rangle$) will only be preserved if the difference parameter is zero.

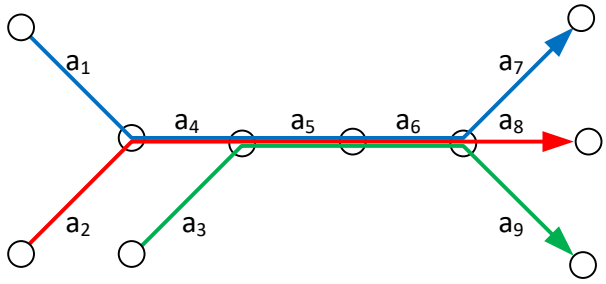


Figure 4: Three different Hot Paths sharing some arcs.

5. DRIVER PROFILE

With the basic definitions of Hot Paths completed, it is time to define how we model a driver profile. We do this by defining the data needed, and thereafter describe the data used for the driver profiles.

5.1 Data Foundation

To be able to find both Hot Paths and driver style we need to have data from different peoples driving. Essentially the bare minimum needed is raw GPS logs with a timestamp and location as the rest can be computed as shown in [19]. We have however built our framework on map matched data, and the data required is listed in Table 1.

Data	Explanation
Locations	The GPS coordinates for each trip.
Timestamps	The time for each location.
Vehicle keys	Unique identifiers for the vehicles.
Arcs	The id and order of the arcs being traversed.
Trips	A unique identifier for each trip made by a vehicle, including location, timestamps and arcs.
Distances on arcs	The distance a driver has driven on the current arc at the time of recording.

Table 1: The data required by our framework.

5.1.1 *Derived Data*

In addition to the data provided by the dataset there are numerous metrics we derive from the original data, e.g. lateral acceleration.

Lateral Acceleration

The lateral acceleration of a vehicle is caused by the force with which it turns. It can be calculated as seen in Equation 11, where v is the velocity and r is the radius of the circle (corner) [2]. Lateral acceleration is interesting to calculate because it gives an indicator of how aggressive drivers are when turning, and this is not covered by any of the other metrics. Furthermore, we can see that insurance companies using telematics are interested in the lateral acceleration [25, 16].

$$a_r = \frac{v^2}{r} \tag{11}$$

To find the radius, we use 3 consecutive GPS points forming a triangle to calculate the circumcircle, and thereby the radius of the turn, as shown in Equation 12, where a, b, and c denote the length of the triangle sides and K is the area of the triangle.

$$r = \frac{abc}{4K} \tag{12}$$

Wobble

When a driver does not pay attention to the road and starts to drift towards the side or center of the road, the driver makes corrections to the direction of the car. Instead of driving in a straight line, the driver ends up driving in a slalom-like pattern. We use the term *wobble* as a measure of these corrections. When a driver pays attention to the road and is driving normally there should only be a small amount of wobble, however when a driver is not focused, the amount of wobble will increase. To calculate wobble we develop an equation that utilize the bearing and speed of a vehicle along with the *bearingdiff* function defined in [19], modified to use degrees:

$$\begin{aligned} bearingdiff\,(\theta_1, \theta_2) = ((\theta_1 - \theta_2 + 180) \bmod 360 + 360) \\ \bmod 360 - 180 \end{aligned} \tag{13}$$

Where θ_1 and θ_2 are the heading of the previous and current GPS record respectively. Using the bearingdiff and speed of a vehicle during a given time period we calculate the wobble as follows:

$$\begin{aligned} totalbdiff\,(Tr, n, k) &= \sum_{i=n-k}^{n-1} bearingdiff\,(g_{i+1}.\theta, g_i.\theta) \\ posdiffs(Tr, n, k) &= \sum_{i=n-k}^{n-1} 1, if\, bearingdiff\,(g_{i+1}.\theta, g_i.\theta) > 0 \\ negdiffs(Tr, n, k) &= \sum_{i=n-k}^{n-1} 1, if\, bearingdiff\,(g_{i+1}.\theta, g_i.\theta) < 0 \\ lastspeeds(Tr, n, k) &= \frac{1}{k} * \sum_{i=n-k}^{n-1} g_i.v \end{aligned} \tag{14}$$

$$\begin{aligned} w(Tr, n, k) = totalbdiff\,(Tr, n, k) \\ * \frac{\min(posdiffs(Tr, n, k), negdiffs(Tr, n, k))}{\max(posdiffs(Tr, n, k), negdiffs(Tr, n, k))} \\ * \frac{lastspeeds(Tr, n, k)}{80} \end{aligned} \tag{15}$$

where Tr is the the trajectory, n is the GPS record number and k is the number of GPS records before the current GPS record to be used in the calculation.

The reason for using wobble rather than just looking at differences in bearing is to only capture the slalom-like pattern and not ordinary turns. Furthermore the equation looks

at the $k-1$ last GPS points which makes it possible for the end-user to adjust the setting to the situation, e.g. setting it high when only looking at motorway driving. The speed is also considered when calculating wobble as wobbling at high speed are worse than at low speed.

5.2 Driver Profiles

In order to cluster drivers, we need to define the data representing each driver. We are using data for specific Hot Paths to get a fair comparison between drivers. Based on the data for each trip driven on the Hot Path we generate a driver profile describing the driving behavior of each trip. This includes averages of accelerations, decelerations, and number of times aggressive behavior of different kinds are detected.

Furthermore, additional data parameters can easily be taken into account when clustering. The only requirement for driver profiles with regards to the data is that the first column must be a unique identifier. The number of columns after that does not matter.

The complete list and explanation of parameters used in the driver profiles for this project can be seen in Table 2. All values are aggregated and averaged, except for the counted values, e.g. *num hard acc*. The counted values are incremented each time a value higher than a user-defined threshold is detected, e.g. number of accelerations above 2.2 m/s^2.

Parameter	Description
Acc pos	Average of positive speed changes.
Acc neg	Average of negative speed changes.
Jerk pos	Average of positive acceleration changes.
Jerk neg	Average of negative acceleration changes.
Lat acc	Average of turn force. Equation defined in Section 5.1.1.
Wobble	Average score of slalom-like driving. Equation defined in Section 5.1.1.
Num hard acc	Number of occurrences of *acc pos* above a user-defined threshold.
Num hard dec	Number of occurrences of *acc neg* below a user-defined threshold.
Num hard pos jerk	Number of occurrences of *pos jerk* above a user-defined threshold.
Num hard lat acc	Number of occurrences of *lat acc* above a user-defined threshold.
Num hard wobble	Number of occurrences of *wobble* above a user-defined threshold.

Table 2: Overview and explanation of parameters in driver profiles.

A driver may have driven the Hot Path multiple times on separate trips, as such the driver profiles for each trip can be averaged to get an overall representation of each driver across all trips on the given Hot Path.

5.2.1 Normalization

Since a driver profile contains numerical representations of several very different metrics with varying ranges of possible values, we use normalization. This is a technique useful for bringing different values to a common scale and ensuring certain metrics does not dominate others, e.g. speed versus binary [13]. We have chosen *Z-score* as it is recommended by Han et al. [13] and provides a usable common scale while also being easy to implement, as well as *Min-Max normalization* as it is commonly used for machine learning [30].

Z-score

The Z-score is defined as the distance from the average of the entire set divided by the standard deviation [30]. This results in a score having a zero mean, where $abs(score)$ is the number of standard deviations from the mean. For a dataset having a normal distribution (Bell Curve) approximately 95 % of the data should have a Z-score between -2 and 2.

The equation for computing the Z-score can be seen in Equation 16, where μ is the average of the data points and σ is the standard deviation.

$$z = \frac{(x - \mu)}{\sigma} \tag{16}$$

Min-Max Normalization

A typical normalization technique is the Min-Max normalization [30], which scales the data to a common data range, usually between zero and one. The Min-Max normalized value can be calculated using Equation 17.

$$x_{norm} = \frac{(x - x_{min})}{x_{max} - x_{min}} \tag{17}$$

6. CLUSTERING

In this section we will look into different methods for clustering drivers based on their driver profile. After the clustering methods we will discuss dimensionality reduction on the driver profiles, and lastly different methods for scoring the resulting clusters.

6.1 Clustering Methods

We start out with the simple, but effective, k-means clustering method before moving onto DBSCAN.

6.1.1 k-means

Clustering based on k-means is a popular method for partitioning a set of data points into a number of clusters [28]. In our case with an insurance company, they may have 5 different premiums for car insurances. k-means can then be used for finding 5 different clusters comprised of drivers that are most similar, after which the insurance company can decide on a premium for each cluster.

For clustering with k-means we use a simple implementation that takes the number of clusters, k, and an $N \times M$ matrix as input, where N is the number of drivers and M is the number of parameters mentioned in Section 5.2. The algorithm generates k *centroids* with M dimensions, and assign each data points to its closest centroid. Then each centroid is set to the average of its assigned data points. The algorithm then assigns each point to its closest centroid again. This continues until the clusters does not change or until a specified maximum number of iterations is reached.

The reason for choosing k-means is due to its simplicity making it useful as a baseline algorithm for comparison. k-means is also very suitable for classifying drivers from an insurance company's point of view, as there is usually a specific number of premiums to choose from, which could be a guideline for choosing the value of k.

6.1.2 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based clustering algorithm that groups data points based on their proximity [11].

Instead of simply finding a fixed number of clusters as in k-means, DBSCAN only cluster data points together if there are a certain amount of them within a certain distance. In addition to the data that is being clustered the algorithm takes two parameters: ϵ (distance between data points) and *min samples* (minimum points to start a cluster). An example of DBSCAN can be seen in Figure 5, which will be used for explaining how the algorithm works in the following.

The algorithm starts by picking an unvisited data point and uses ϵ as a search radius to determine the number of points in the vicinity. In the example we see that the point A has 3 other points within its search radius, meaning that if the minimum points variable was set to 4 or lower, a cluster is found. Each new point found is then controlled for points within their search radius. This is the step where the two yellow points (edge points) and the two additional core points (red) are found. Core points have at least $min\ samples - 1$ other data points in the cluster within their search radius where edge points only have a single data point in its search radius.

The process then restarts at a new unvisited point, until all points has been visited.

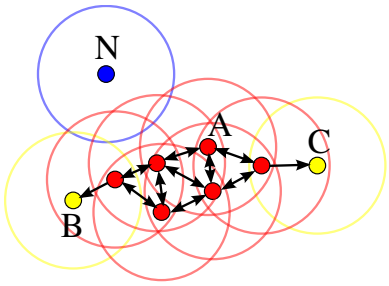


Figure 5: An example of DBSCAN clustering, where core points are illustrated in in red, cluster edge points in yellow, and outliers in blue. Illustration from [6].

DBSCAN was chosen because, just as k-means, it is a relatively simple clustering algorithm, while having an approach that differs greatly from k-means in that it groups based on the proximities of the data points and identifies outliers.

6.2 Dimensionality Reduction

Dimensionality reduction, or rather feature extraction, is useful for high-dimensional datasets, such as our driver profiles, as it reduces the number of dimensions into a low-dimensional representation, e.g. two dimensions. In addition it enables visualization of the data and clusterings, when reducing to 3 or fewer dimensions. A feature of the dimensionality reduction techniques we consider are also that the proximity of the high-dimensional data will also correlate with their proximity in the reduced space. This means that the correlation of data points in the original space is preserved, which is essential when clustering.

In the following we will be looking into two different methods for reducing dimensions, namely Self-Organizing Feature Maps and t-Distributed Stochastic Neighbor Embedding.

6.2.1 Self-Organizing Feature Maps

Self-Organizing Feature Maps (SOFM) is a type of artificial neural network that apply competitive learning [21, 20]. It produces a low-dimensional representation of the input called a map, e.g. reduction to two dimensions. A benefit of SOFM is the unsupervised learning of the artificial neural network, which in our case is ideal as we do not wish to state what is and what is not good driving behavior.

6.2.2 t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is another dimensionality reduction method, comprised of two stages [27]. During the first stage the algorithm constructs a probability distribution over pairs of high-dimensional data such that similar data points have a high probability of being picked, whereas dissimilar points have a small probability. This is similar to SNE, which t-SNE is built upon. During the second stage t-SNE defines a similar probability distribution over the points in an *embedding* (low-dimensional map), and minimizes the Kullback-Leibler divergence [23] between the embedding and the original dataset, which tries to ensure that the embedding most closely represents the original data.

6.2.3 Discussion

Due to performance issues with SOFM in our setup we will not be using it. Reducing driver profile data into two-dimensions with SOFM often requires more than 1 hour, whereas t-SNE in most cases completes in less than 5 seconds.

6.3 Scoring

Since we have no ground truth for evaluating the clusters we have to utilize a more objective method for determining how effective our clustering is. As such we look into scoring methods for evaluating clusters.

From the Python library "scikit-learn," used for some of our clustering methods, we already have access to two scoring methods that are not based on matching the clustering result with the truth (classification) [32]. These scoring methods are the Calinski and Harabasz score and Silhouette score. However, since both of these scoring methods are better for convex clusters rather than density based clusters, e.g. clusters made with DBSCAN [7], we have to include additional scoring methods. As such we additionally look into Davies-Bouldin and Dunn index.

The framework makes it easy to use additional scoring methods, which is important as different scoring methods perform differently according to the given data.

All of the methods mentioned here are methods that output a numerical score that describe quality of the cluster in terms of the proximity of data points within the cluster, distance between clusters, and so on. These scores are also relatable to telematics as a better score means that the different groups are easier to distinguish, making the cluster better for grouping driving styles.

6.3.1 Calinski and Harabasz Score

The Calinski and Harabasz score [4] is defined as a ratio between the dispersion of elements within a cluster against the dispersion of clusters, where a higher value is preferred. The equation for calculating the score can be seen in Equation 18, where k is the number of clusters, n is the total number of points, n_i is the number of points in cluster i, $\overline{d}^2$ is the general mean of squared distances and $\overline{d_i}^2$ is the mean of squared distances within cluster i.

$$C = \frac{BGSS}{k-1} \; / \; \frac{WGSS}{n-k}$$
$$\text{where } WGSS = \frac{1}{2} * \sum_{i=1}^{k} \left((n_i - 1) * \overline{d_i}^2 \right) \tag{18}$$
$$, \; BGSS = \frac{1}{2} * \left((k-1)\,\overline{d}^2 + (n-k)\,A_k \right)$$

6.3.2 Davies-Bouldin Index

The Davies-Bouldin index [9] is an expression of the ratio between distances within a cluster to the distance between clusters. The numerical output is above 0, where the lower the value, the better the clustering is.

The algorithms for calculating the Davies-Bouldin Index are listed in the equations from Equation 19 to 23. S_i, in Equation 19, is a measure of scatter within a given cluster. Furthermore, T_i is the number of data points within the cluster, X_j is a data point, and A_i is the centroid of the given cluster. Finally we have p, which usually is set to 2 making it a Euclidean distance function. This has to match the distance function used when doing the clustering.

$$S_i = \left(\frac{1}{T_i} \sum_{j=1}^{T_i} |X_j - A_i|^p \right)^{\frac{1}{p}} \tag{19}$$

$M_{i,j}$, in Equation 20, is a measure of the distance between two clusters. $a_{k,i}$ is the kth element of A_i.

$$M_{i,j} = ||A_i - A_j||_p = \left(\sum_{k=1}^{n} |a_{k,i} - a_{k,j}|^p \right)^{\frac{1}{p}} \tag{20}$$

Since Davies-Bouldin is a ratio of the distance between data points within a cluster to the distance between clusters, it is exactly what is calculated in Equation 21.

$$R_{i,j} = \frac{S_i + S_j}{M_{i,j}} \tag{21}$$

The final two equations, Equation 22 and Equation 23, utilize the declared equations to put it all together.

$$D_i \equiv \max_{j \neq i} R_{i,j} \tag{22}$$

$$DB \equiv \frac{1}{N} \sum_{i=1}^{N} D_i \tag{23}$$

6.3.3 Dunn Index

The Dunn index [10] is the ratio of the smallest distance between any two data points of two different clusters to the largest distance between any two data points within the same cluster. This is between 0 and ∞, where the higher the value the better the clustering is.

The definition for Dunn index is in Equation 24, where $\delta(C_i, C_j)$ is the distance between the clusters C_i and C_j and $\triangle i$ is the maximum distance between two points within the cluster C_i.

$$\triangle i = \max_{x,y \in C_i} d(x,y)$$
$$DI_m = \frac{\min\limits_{1 \leq i < j \leq m} \delta(C_i, C_j)}{\max\limits_{1 \leq k \leq m} \triangle k} \tag{24}$$

6.3.4 Silhouette Score

The silhouette score [31] measures cohesion (how similar elements in a cluster are to each other), where the score ranges from -1 to 1. The higher the score, the better the clustering is and elements of different clusters will be dissimilar. Scores around 0 indicate overlapping clusters, and negative scores indicates that elements have been wrongly assigned.

Silhouette score is defined in Equation 25, where $a(i)$ is the average distance of i to all other data points within its cluster and $b(i)$ is the lowest average distance of i to any cluster other than its own.

$$s(i) = \frac{b(i) - a(i)}{\max a(i), b(i)} \tag{25}$$

7. EXPERIMENTS

This section will outline the experiments we have performed to assess the performance of our framework. We begin by describing the data used for the experiments before detailing the performance experiments.

7.1 Data Foundation

The experiments made in this paper are based on the ITS [17] dataset, which is a 217 GB dataset collected in 2012-2014, containing map matched GPS trajectories from 458 vehicles in Denmark. Some experiments are focusing on specific geographical areas and other filters. The dataset and selected filters are summarized in Table 3.

The ITS dataset supports SPQ defined in [22], usable for path-based analysis.

7.2 Hot Paths

In this section we show our Hot Path experiments, where we start by investigating minimal support values, before doing performance measurements, and finally showing the number of Hot Paths discovered.

7.2.1 Investigating Minimal Support Values

To extract Hot Paths we use the ITS dataset with some initial filtering:

- Limit to the Aalborg/Nørresundby city area.
- We exclude the road categories: residential, living street, unpaved, service, unclassified. These are not likely Hot Paths as they are of little importance to traffic in general. We exclude these as the data is biased to a small area, which might result in erroneous

Filter	Item	Count
None	Data points	1,144,334,515
	Trips	1,381,021
	Vehicles	458
	Arc passages	79,434,563
	Used arcs	391646
Aalborg / Nrsb	Data points	275,051,868
	Trips	520,230
	Vehicles	424
	Arc passages	21,541,883
	Used arcs	13,269
Aalborg / Nrsb excluding minor roads	Data points	227,222,840
	Trips	508,023
	Vehicles	424
	Arc passages	17,414,841
	Used arcs	2,594
Aalborg / Nrsb excluding minor roads and first and last 500m of each trip	Data points	202,004,086
	Trips	471,691
	Vehicles	424
	Arc passages	15,216,999
	Used arcs	2,594

Table 3: Summation of specific filters used. Rows 2-4 constitute subsets of the preceding rows.

Hot Paths. As for residential and living street, they do not take almost no part in any Hot Path, even if trip support is as low as 5,000 as illustrated in Figure 7.

- We remove the first and the last 500 m of each trip due to privacy concerns.

The data is limited to Aalborg/Nørresundby as the data is heavily biased towards this region of Denmark, as seen in Figure 6, as such this have no effect on the general applicability of the method.

This leaves the following amount of data for the Hot Path extraction:

- 1,868 segments, but only 1,805 has data in the ITS dataset.
- 1,628,110 segment pairs for the SPQ algorithm before trip support filtering, but only 249,671 pairs if $S_{trip} \geq 10,000$.

When saving the Hot Paths, we only keep Hot Paths with these properties:

- Contains at least 5 arcs.
- Has a length of at least $500\,m$.

The cut-off values for these properties can of course be changed, however we choose these limitation as Hot Paths with smaller lengths and less arcs will only have a small amount of data for each trip.

The effect of a trip support limit on the entire dataset can be seen in Figure 7, where the x-axis shows the limit on trip support and the y-axis shows how many percent of the total

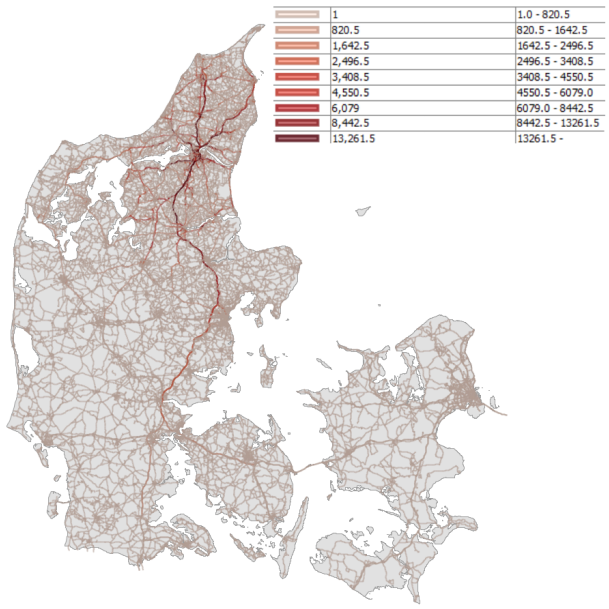


Figure 6: Heatmap of the ITS dataset.

arcs are left for each road type. The figure shows, that when the minimum trip support is chosen to a value above 10,000, almost all road types except motorways are removed. These values will of course differ for different datasets or if limitations are used on the dataset, e.g. looking at data for another city, or only considering specific time periods.

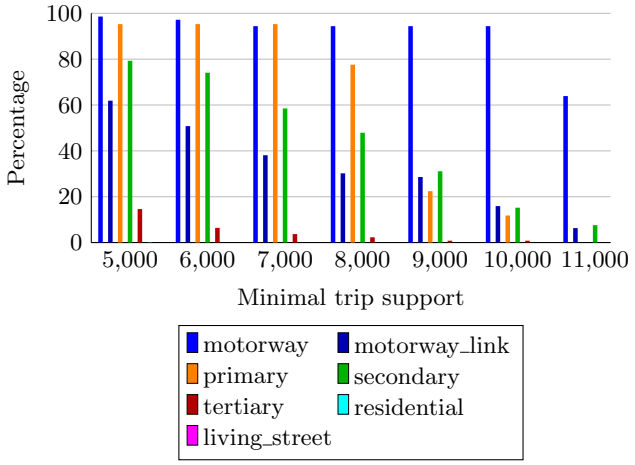


Figure 7: Percentage of arcs being part of a Hot Path for different road types for different support values.

7.2.2 *Hot Path Performance Measurements*

We evaluate the performance of our suggested Apriori algorithm by comparing it to the performance of the implementation based on SPQ, and the Graph Based (GB) implementation. The results can be seen in Figure 10, which shows the runtime of the different algorithms for different trip support values. For the Apriori and GB, the tests are done setting vehicle support to zero (which is not the best case). The SPQ based implementation is minimal, and does

not include handling of context or vehicle support, we include the previously mentioned filters with Hot Paths of at least 500 m.

Finally, as mentioned in Section 4.2, SPQ suffers a serious performance hit when having to look up data for the actual trips, as such to uphold this limitation of Hot Paths being at least 500 m, and as we remove the first and last 500 m of each trip, we only include segment pairs being at least 400 m apart in the SPQ Hot Path (SPQHP) generation.

These changes have been made in order to alleviate some of the performance problems SPQHP has, and only boosts the performance of SPQHP.

The results in Figure 10 shows that the implementation using Apriori is faster than the one using SPQHP, which is likely to become more prominent if we were to add handling of context or vehicle support when using SPQHP. Apriori is also faster than GB. The SPQHP implementation is reasonably fast when the number of segments is low, but becomes substantially slower as the number of segments increase. The GB implementation performs even worse than the SPQHP implementation when trip support is high, however it should be noted that the GB implementation considers contexts, whereas SPQHP does not. In contrast, the Apriori implementation scales better and is less sensitive to larger amounts of data.

These results are not surprising, as the work done by the SPQHP implementation is proportional to the number of map segment combinations, i.e. $O(N^2)$, while the work done by the Apriori implementation is expected to be dominated by sorting, i.e. $N * log(N)$ where N is the number of *arc passages* (times an arc is passed by a trip).

Note that the performance of the Apriori implementation only vary little as the minimal vehicle support is changed. This is due to trip support and vehicle support being correlated, as illustrated in Figure 8. Vehicle support only makes a difference when trip support is low. This can be seen in Figure 9, where varying vehicle support values only make a significant difference when trip support is low (5,000), or when vehicle support is very high (350). This intuitively makes sense, as more vehicles are needed to achieve a higher trip support.

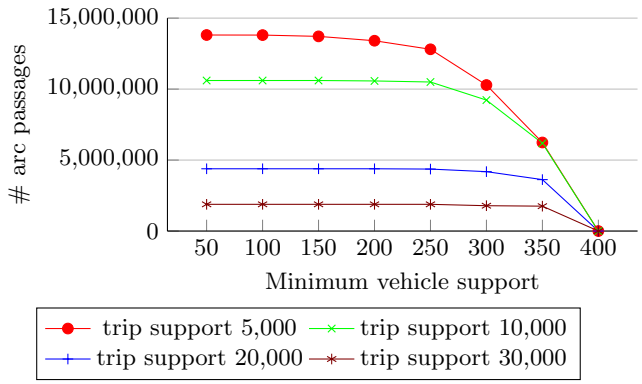


Figure 8: Correlation between S_{trip} and $S_{vehicle}$.

When implementing the Apriori algorithm, we have a choice of when to perform filtering based on vehicle support. It can be done early i.e. when generating the itemsets, or late i.e. when saving the results. Early filtering is only the fastest option when minimum trip support is low and minimum vehicle support is high, as can bee seen in Table 4 . The numbers used for Figure 9 is from the fastest of the two options.

Filter	S_{trip}	Minimum S_{trip} 100	200	300	350
Early	5,000	2750 s	2722 s	1382 s	554 s
	10,000	909 s	907 s	758 s	481 s
	20,000	280 s	279 s	272 s	238 s
	30,000	158 s	160 s	158 s	158 s
Late	5,000	2275 s	2281 s	2170 s	2069 s
	10,000	700 s	704 s	701 s	682 s
	20,000	195 s	197 s	193 s	191 s
	30,000	105 s	103 s	104 s	101 s

Table 4: Runtime comparison of early vs. late vehicle support filtering for Hot Path generation using the Apriori algorithm. Green numbers indicate the fastest, and red the slowest.

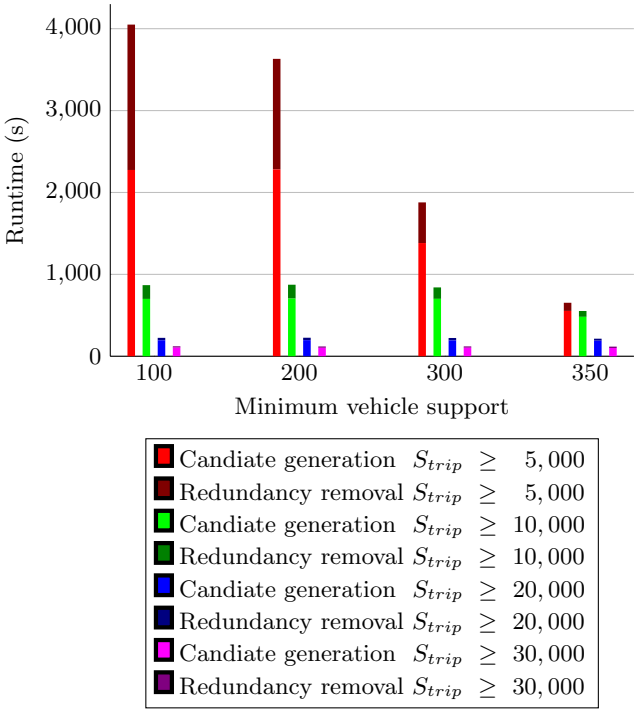


Figure 9: Runtimes for Apriori Hot Path extraction Aalborg/Nørresundby area. The *difference* parameter for the reduction step is set to 0.5.

7.2.3 Hot Paths Discovered

The Hot Paths found when minimum trip support is 5,000 and 10,000 can be seen in Figure 11. The black circle at the center illustrates an interesting spot, where a Hot Path is "broken", somewhat surprisingly. The reason for this particular gap is, that at the eastern end, the traffic is split into two roads, and at the western end, part of the traffic is directed into the parking lot of a big grocery store.

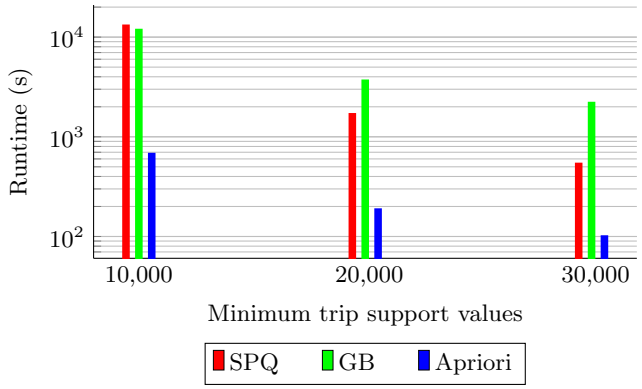


Figure 10: Runtime for SPQ and GB Hot Path generation for Aalborg/Nørresundby (without reduction step).

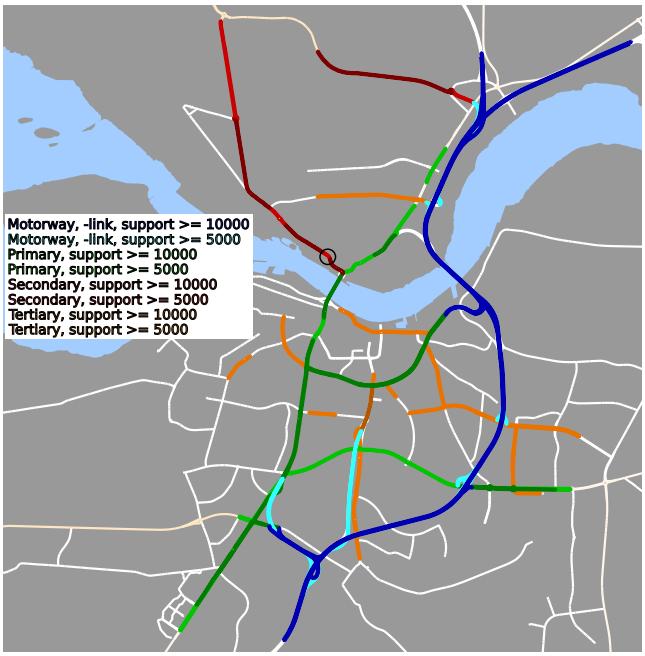


Figure 11: Hot Paths with more than 5.000 in support in Aalborg, Denmark.

7.3 Clustering

This section describes the results of running the different clustering algorithms mentioned in Section 6 on different Hot Paths. In total we have four different combinations, k-means and DBSCAN without dimensionality reduction, and k-means and DBSCAN with dimensionality reduction through t-SNE. Furthermore, because there are numerous different settings for the three algorithms (DBSCAN, k-means, and t-SNE) we will be utilizing grid search [14] to find the optimal settings.

7.3.1 Grid Search

Due to a lack of ground truth we utilize the clustering scores described in Section 6.3. We choose the highest scoring candidate for each cluster score, resulting in up to 4 candidates for each clustering method. From these candidates the one with the overall best score is chosen as having the optimal settings for this method and data.

In order to be able to compare results from k-means and DBSCAN we would like to use the same embedding for both clustering methods.

When choosing the optimal settings for DBSCAN we have to consider that DBSCAN can identify points as outliers. This is troublesome for our cluster scoring methods as these only consider the points marked as clusters, meaning that clusterings with large amounts of outliers can result in good clustering scores, which is not optimal. As such we place a user-defined restriction on the DBSCAN cluster results that we want at most 25 % outliers in the resulting clusters. The parameters being tuned in the grid search can be seen in Table 5. It is worth mentioning that both the range and increment are user-defined, and can be altered according to the dataset or runtime requirements (larger ranges and smaller increments results in a longer runtime).

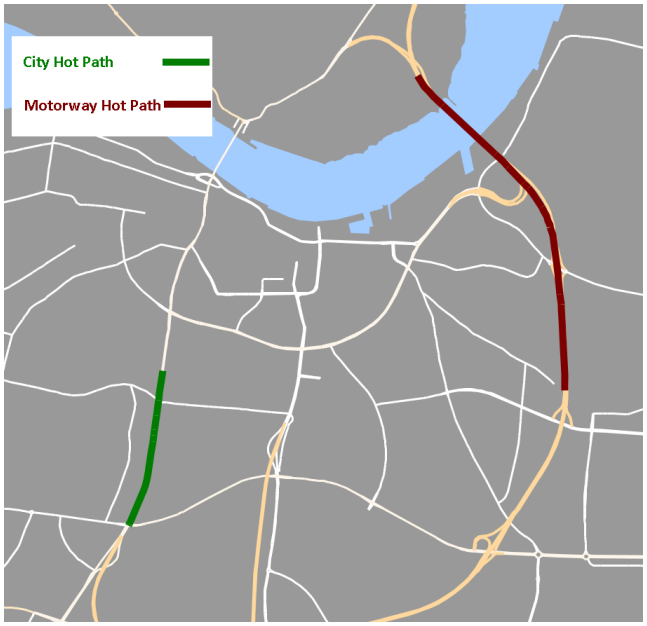


Figure 12: The two chosen Hot Paths for clustering evaluation.

Algorithm	Setting	Range		Increment
DBSCAN	ϵ	0.033	7.986	0.033
k-means	Num clusters	2	10	1
t-SNE	Learning rate	4	24	1
	Perplexity	20	32	1

Table 5: Parameters that are varied during the grid search.

7.3.2 Data for Clustering

We cluster on two different Hot Paths, one on the E45 motorway (Motorway Hot Path) and one found on Hobrovej (City Hot Path), one of the larger city roads in Aalborg. The Hot Paths can be seen in Figure 12. These two Hot Paths have been chosen at random with the only constraint being to represent two different road types and use, i.e. city road and motorway.

For the clustering experiments the context used on the Hot Paths is from June to August, Wednesdays from 7 A.M. to 8:20 A.M. For simplicity the context for the experiments are defined using the first arc of each path. This gives us 124 trips in total from 36 drivers for City Hot Path and 429 trips in total from 90 drivers on Motorway Hot Path.

The data used is not normalized. The effect of normalization is explored in Section 7.3.5.

7.3.3 Clustering Results

In this section we present the results of our clustering methods.

k-means

The settings and runtime of the k-means algorithm for City Hot Path and Motorway Hot Path can be seen in Table 6. The average values of the resulting clusters for City Hot Path can be seen in Table 11 and the ones for Motorway Hot Path can be seen in Table 14. We find that k-means is very fast at creating clusters, making the method usable in real time if optimal settings are known beforehand, e.g. if an insurance company has a set number of premiums.

We see that the data for City Hot Path has 6 clusters as compared to the 2 found on Motorway Hot Path, which could indicate that there is a larger difference in driving style when users drive on city roads compared to motorways.

Hot Path	Clusters	Iterations	Runtime
City	6	500	37 ms
Motorway	2	500	39 ms

Table 6: Settings and runtime of the k-means algorithm.

DBSCAN

The settings and runtime of the DBSCAN algorithm for Motorway Hot Path can be seen in Table 7. The average values of each resulting cluster can be seen in Table 15.

A meaningful clustering for City Hot Path was not obtainable by DBSCAN as the number of outliers were either above 25 % or only one cluster was found.

We see that DBSCAN has a significantly higher runtime than k-means, but still within a few seconds. The number of clusters coincide with the number k-means have found, also indicating that there are less variations in driving styles for Motorway Hot Path.

Hot Path	Clusters	ϵ	Min Samples	Runtime
Motorway	2	7.029	5	1534 ms

Table 7: Settings and runtime of the DBSCAN algorithm on data from Motorway Hot Path.

t-SNE

The settings found using grid search and the runtime of the t-SNE algorithm for City Hot Path and Motorway Hot Path can be seen in Table 8. Settings and runtime for k-means and DBSCAN using t-SNE can be seen in Table 9 and 10 respectively. The result of clustering the two-dimensional data generated by t-SNE can be seen in Figure 13 for City Hot Path, and in Figure 14 for Motorway Hot Path. The average values of each resulting cluster can be seen in Table 12 and 13 for City Hot Path and Table 16 and 17 for Motorway Hot Path.

We see that using t-SNE dimensionality reduction changes the clusterings significantly. For City Hot Path we see that using t-SNE with k-means the best clustering has 2 clusters opposed to 6 clusters for the raw driver profiles. DBSCAN on the other hand is now able to find a proper clustering for 8 clusters, which is in agreement with the number of clusters found by k-means on the raw driver profiles.

For Motorway Hot Path we see that the number of clusters found using k-means is 8 using t-SNE and only 2 using the raw driver profiles. This is the reverse of what could be concluded with just k-means.

DBSCAN, on the other hand, is in agreement with the results found using the raw driver profiles. We do see that an additional small cluster has also been found, which could indicate that t-SNE was able to more clearly reveal variations in the driving styles.

Hot Path	Perplexity	Learn rate	Runtime
City	32	15	1068 ms
Motorway	27	22	4126 ms

Table 8: Settings and runtime of the t-SNE algorithm.

Hot Path	Clusters	Iterations	Runtime
City	2	500	21 ms
Motorway	8	500	55 ms

Table 9: Settings and runtime of the k-means algorithm using t-SNE.

Hot Path	Clusters	ϵ	Min samples	Runtime
City	8	0.759	5	112 ms
Motorway	3	0.594	5	736 ms

Table 10: Settings and runtime of the DBSCAN algorithm using t-SNE.

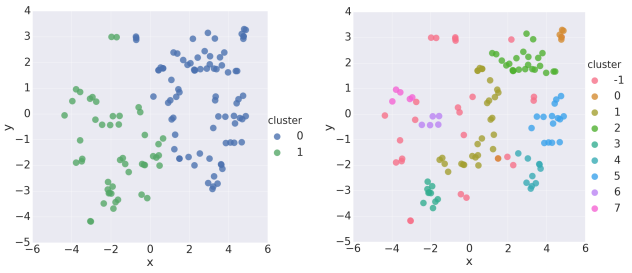
7.3.4 Evaluating Clusters

As we do not have a ground truth for the data, evaluating the exact clusterings is not possible, however we can objectively look at the data to try and assess the driving style of each cluster.

City Hot Path

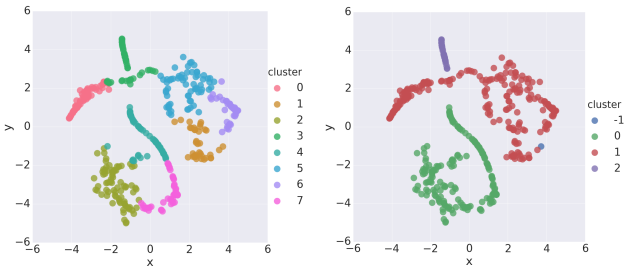
The results from k-means in Table 11 show 6 different clusters, where 2 of them (cluster 0 and 3) consists of 90 % of the driver profiles. Cluster 0 and 2 appear to be similar when disregarding lateral acceleration. These clusters appear to represent calm driving styles. Cluster 1, 3, 4, and 5 all seem to have varying degrees of aggressive driving style. This is also apparent in Figure 15, where we see each clusters distribution with regards to positive and negative acceleration. In Section A similar plots regarding lateral acceleration and jerk can be seen.

Looking at Figure 13a, showing the clustering on City Hot Path using k-means and t-SNE, we can identify two distin-



(a) k-means (b) DBSCAN

Figure 13: The t-SNE embedding from City Hot Path that achieved highest cluster scores with k-means and DBSCAN collectively.



(a) k-means (b) DBSCAN

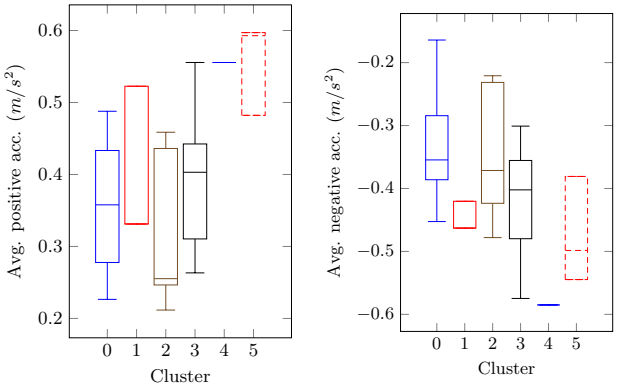
Figure 14: The t-SNE embedding from Motorway Hot Path that achieved highest cluster scores with k-means and DBSCAN collectively.

guishable clusters. Looking at the data for the clustering on City Hot Path, seen in Table 12, we can easily classify the two clusters as calm (cluster 0) and aggressive (cluster 1) drivers. This correlates with what we saw when just using k-means.

The t-SNE and DBSCAN clustering seen in Table 13 has a total of 8 clusters. Cluster 1, which is the cluster in the middle of Figure 13b, is the most average cluster, probably due to its size and distance between elements. Cluster 0, 2, 4, and 5, which are all on the right side, have similar values in *jerk pos* and *jerk neg*, somewhat in *acc pos* and *lat acc pos*, but dissimilar on most of the others. This behavior coincides with the calm driving style from the two earlier methods. We also see that cluster 6 and 7 are quite similar on most parameters, but differs on jerk and lateral acceleration. It is clear that most of these clusters have their traits, however they could be classified as being similar, e.g. cluster 3, 6, and 7, belonging to an "aggressive" cluster due to their high lateral acceleration and wobbling. Even though there are more clusters when using t-SNE and DBSCAN, they are not conflicting with the other methods. The resulting clusters are simply more specific in their individual traits, but their general similarity remains.

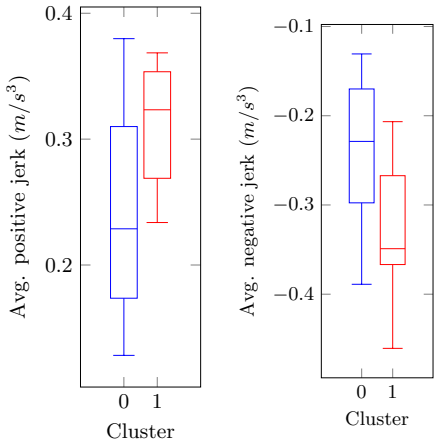
Motorway Hot Path

The results for both k-means and DBSCAN, seen in Table 14 and 15 respectively, show an agreement on number of clusters, their sizes, and the driving style for each cluster. Even though cluster 0 have more than 95 % of all driver profiles in both cases we see that it has a more calm driving style compared to the total average on all parameters. How



(a) Acceleration positive. (b) Acceleration negative.

Figure 15: City Hot Path acceleration using k-means.



(a) Jerk positive. (b) Jerk negative.

Figure 16: Motorway Hot Path jerk using DBSCAN.

the driving styles for jerk are distributed for DBSCAN can be seen in Figure 16. The clustering using k-means and DBSCAN with t-SNE are similar, the distributions for these can be seen in Section A, Figure 19 and 20.

The result for t-SNE and k-means seen in Table 16 is similar to the result we saw in City Hot Path with t-SNE and DBSCAN where additional clusters display more specific individual traits of each cluster, but they seem to capture the same general driving styles.

Looking at the data from t-SNE and DBSCAN in Table 17 and the visual representation of the data in Figure 14b, it is clear to see that DBSCAN have captured the 3 distinct groups in the data. We also see that the clusterings found by DBSCAN with t-SNE quite closely resembles the clusterings found by k-means and DBSCAN without using t-SNE. We see that they all share a small cluster of drivers with aggressive driving style, but the one calm cluster found in the raw driver profiles have been split in two by DBSCAN when using t-SNE. The data indicates that this split is meaningful as we see a clear distinction in the values for wobble as well as all counted values being high in cluster 0 and low in cluster 1, while the acceleration and jerk are more similar for both clusters, which could explain why they were clustered as one using the raw driver profiles.

cluster	# elements	acc pos	acc neg	jerk pos	jerk neg	lat acc pos	wobble	# hard acc	# hard dec	# hard lat acc	# hard wobble
0	74	94.96	89.29	85.12	84.61	69.15	73.95	59.06	61.29	0	0
1	2	109.98	117.80	193.10	189.27	83.70	216.55	255.91	167.74	0	1250.00
2	6	85.95	87.39	89.98	91.32	313.79	119.23	34.12	83.87	1410.85	83.33
3	40	107.09	118.49	119.98	120.30	116.13	131.03	156.10	150.97	75.58	187.50
4	1	143.12	156.04	259.35	297.56	467.42	319.08	307.09	251.61	1209.30	1000.00
5	3	143.58	126.61	125.24	121.50	147.85	174.41	375.33	363.44	100.78	500.00

Table 11: Averages of each cluster compared to the total average in percent, from clustering using k-means on City Hot Path.

cluster	# elements	acc pos	acc neg	jerk pos	jerk neg	lat acc pos	wobble	# hard acc	# hard dec	# hard lat acc	# hard wobble
0	77	97.16	93.53	88.74	90.00	71.02	76.23	60.66	64.20	0	0
1	49	104.70	110.70	118.61	116.54	147.91	139.30	165.03	159.18	265.31	265.31

Table 12: Averages of each cluster compared to the total average in percent, from clustering using k-means and t-SNE on City Hot Path.

cluster	# elements	acc pos	acc neg	jerk pos	jerk neg	lat acc pos	wobble	# hard acc	# hard dec	# hard lat acc	# hard wobble
0	5	90.30	40.23	77.52	87.86	49.28	92.52	57.22	53.09	0	0
1	26	98.77	110.15	101.86	102.35	104.45	110.95	73.57	178.24	0	0
2	24	95.41	89.28	91.45	90.54	71.17	73.28	0	0	0	0
3	10	109.04	120.16	149.38	144.06	138.26	209.61	154.50	84.95	0	965.63
4	10	97.55	104.58	82.30	79.97	50.41	55.90	103.00	276.08	0	0
5	15	114.13	107.57	90.06	90.20	56.13	64.78	297.56	28.32	0	0
6	5	96.71	103.63	125.55	126.06	169.70	121.76	103.00	63.71	436.97	0
7	5	88.30	90.51	98.54	102.15	358.91	128.78	68.67	127.42	1623.03	128.75

Table 13: Averages of each cluster compared to the total average in percent, from clustering using DBSCAN and t-SNE on City Hot Path.

cluster	# elements	acc pos	acc neg	jerk pos	jerk neg	lat acc pos	wobble	# hard acc	# hard dec	# hard lat acc	# hard wobble
0	403	98.06	97.43	97.77	97.19	99.58	72.75	89.60	92.36	95.09	93.79
1	26	130.01	139.79	134.62	143.61	106.45	522.37	261.15	218.38	176.07	196.24

Table 14: Averages of each cluster compared to the total average in percent, from clustering with k-means on Motorway Hot Path.

cluster	# elements	acc pos	acc neg	jerk pos	jerk neg	lat acc pos	wobble	# hard acc	# hard dec	# hard lat acc	# hard wobble
0	379	99.53	99.31	99.75	99.60	99.93	97.63	99.67	98.35	95.50	99.16
1	6	129.89	143.85	115.97	125.07	104.28	249.55	121.07	204.24	384.52	153.23

Table 15: Averages of each cluster compared to the total average in percent, from clustering with DBSCAN on Motorway Hot Path.

cluster	# elements	acc pos	acc neg	jerk pos	jerk neg	lat acc pos	wobble	# hard acc	# hard dec	# hard lat acc	# hard wobble
0	61	60.15	53.79	61.98	59.48	94.90	0.07	5.06	0	0	0
1	37	110.89	94.40	94.88	95.77	91.55	67.31	133.46	73.08	113.12	90.06
2	77	94.16	100.04	95.43	97.76	99.67	129.56	88.18	115.88	98.52	216.37
3	47	108.08	109.07	117.06	122.49	126.60	307.48	144.47	120.81	116.88	112.99
4	54	133.57	130.68	116.64	115.99	94.60	129.02	188.61	191.94	210.72	177.40
5	64	117.44	141.16	142.21	135.60	111.58	34.17	106.09	153.50	138.97	6.51
6	43	79.99	69.65	73.88	75.98	83.75	40.44	0	6.29	3.04	0
7	46	100.63	91.53	90.90	92.04	92.34	110.55	154.32	111.68	113.73	165.24

Table 16: Averages of each cluster compared to the total average in percent, from clustering the t-SNE data with k-means on Motorway Hot Path.

cluster	# elements	acc pos	acc neg	jerk pos	jerk neg	lat acc pos	wobble	# hard acc	# hard dec	# hard lat acc	# hard wobble
0	177	108.01	107.06	100.63	101.80	96.19	124.91	135.69	137.67	136.39	190.75
1	225	90.21	89.87	95.52	93.55	102.26	31.38	53.37	56.75	62.63	17.54
2	26	130.18	139.64	134.48	143.56	106.42	524.27	260.54	217.87	175.66	195.78

Table 17: Averages of each cluster compared to the total average in percent, from clustering the t-SNE data with DBSCAN on Motorway Hot Path.

Scoring

Now that we have generated clusters it is time to look at how they score. We do this by using the four techniques mentioned in Section 6.3, namely Calinski and Harabasz, Davies-Bouldin, Dunn, and Silhouette. The results can be seen in Table 18 for City Hot Path and in Table 19 for Motorway Hot Path.

Method	Calinski & Harabasz	Davies-Bouldin	Dunn	Silhouette
k-means	59.54862	0.85230	0.10201	0.37630
t-SNE & k-means	43.51493	1.35514	0.04446	0.36231
t-SNE & DBSCAN	49.01549	1.09186	0.07955	0.22681

Table 18: Cluster scoring of the different clusters generated on City Hot Path.

Method	Calinski & Harabasz	Davies-Bouldin	Dunn	Silhouette
k-means	834.31867	0.50499	0.03941	0.84654
DBSCAN	113.73104	0.26856	0.16922	0.69897
t-SNE & k-means	39.91432	1.40050	0.00088	0.15925
t-SNE & DBSCAN	552.18193	0.69665	0.00666	0.47580

Table 19: Cluster scoring of the different clusters generated on Motorway Hot Path.

7.3.5 Adjusting the Dataset

The data used for both City Hot Path and Motorway Hot Path has up until this point been used "as-is", therefore we will now look into what normalization does to it as well as determine what effect the derived data we added does.

Normalization

Normalization has an affect on the data, such that a single dimension is less likely to dominate the entire driver profile. Therefore we ran the clustering with normalization, and interestingly enough found that the two Hot Paths reacted differently to the two methods described in Section 5.2.1. City Hot Path achieved best results with Z-score and Motorway Hot Path achieved it with Min-Max. The results can be seen in Table 20 and 21, respectively.

From this we can see that for our dataset the normalization is somewhat hit-and-miss. In our opinion this is because some of the parameters we have used are somewhat similar in scale to normalized data. Of course this may just be a lucky coincidence. For other parameters normalization removes an important difference in scale.

Furthermore we see that the Dunn-index scoring achieves remarkable results from normalization. As described in Section 6.3.3, the Dunn-index prefers compact clusters with great distance to other clusters, which means that normalization distributes the data in a way where our clustering methods can better create compact and separated clusters.

Derived Data

Before clustering we added two derived values for each data entry, wobble and lateral acceleration. As this is optional data we added on top of the existing data we would like to evaluate whether the clustering actually benefits from this extra data. The cluster scorings from running the clustering without the derived data can be seen in Table 22 for City Hot Path and Table 23 for Motorway Hot Path.

Similar to what we saw with normalization, the cluster scorings with the dataset without derived data displays worse scores for some methods, and improvements for others. Namely k-means for City Hot Path, and k-means with t-SNE for Motorway Hot Path shows better results without derived data.

Overall it is clear to see the derived data has a positive impact for the cluster scoring.

8. CONCLUSION

We present an effective, efficient, and highly flexible framework to generate Hot Paths using a novel variant of the Apriori algorithm and fairly group and evaluate driving styles based on GPS data. Our Apriori variant clearly outperforms the tested alternatives.

Using the data found on Hot Paths the framework is able to group driving styles using k-means and DBSCAN in an optional combination with the t-SNE dimensionality reduction method. We found that the framework finds meaningful clusters showing differences in groups of drivers, such as aggressive and calm driving behavior. We have also shown that these results can be obtained in near-realtime, which is beneficial for end-users experimenting with the clustering.

The framework can also be used for analyzing road networks, as we found that the generated Hot Paths tend to capture the general flow and behavior of drivers in a road network, making them more apparent.

9. FUTURE WORK

In this section we discuss further work that could have improved our framework.

9.1 Classification

For future work it would be interesting to obtain labeled data from an insurance company, having such data would enable us to better evaluate the effectiveness of our framework while also enabling us to benefit from supervised learning in our implementations. Another interesting aspect for future work would be to use more advanced classification methods such as neural networks and Support Vector Machines. These methods would also benefit greatly from having labeled data available.

9.2 Normalization

As indicated in Section 7.3.5 the clustering methods is likely to benefit from the ability to supply weights to normalized data. It could be interesting to explore the effect of such weights. Weighting could also be an interesting feature for a domain expert who has a better understanding of the importance of each feature with regards to a specific domain, e.g. insurance company or fleet owner.

Method	Calinski & Harabasz	Diff.	Davies-Bouldin	Diff.	Dunn	Diff.	Silhouette	Diff.
k-means	48.32880	81.16 %	0.85381	100.18 %	0.44439	435.63 %	0.70538	187.45 %
t-SNE & k-means	30.62817	70.39 %	1.70986	126.18 %	0.05940	133.61 %	0.29204	80.61 %
t-SNE & DBSCAN	39.64217	80.88 %	1.13758	104.19 %	0.31714	398.69 %	0.64021	282.27 %

Table 20: Cluster scoring of the different clusters generated with Z-score normalized data on City Hot Path. The percentage is compared to the scores in Table 18.

Method	Calinski & Harabasz	Diff.	Davies-Bouldin	Diff.	Dunn	Diff.	Silhouette	Diff.
k-means	204.91672	24.56 %	1.29118	255.69 %	0.05687	144.31 %	0.33022	39.01 %
t-SNE & k-means	136.07694	340.92 %	1.25603	89.68 %	0.06020	6849.64 %	0.22811	143.24 %
t-SNE & DBSCAN	147.63848	26.74 %	1.16499	167.23 %	0.06020	904.51 %	0.22358	46.99 %

Table 21: Cluster scoring of the different clusters generated with Min-Max normalized data on Motorway Hot Path. The percentage is compared to the scores in Table 19.

Method	Calinski & Harabasz	Diff.	Davies-Bouldin	Diff.	Dunn	Diff.	Silhouette	Diff.
k-means	90.66014	152.25 %	0.51752	60.72 %	0.54051	529.86 %	0.79055	210.08 %
t-SNE & k-means	34.92149	80.25 %	1.37833	101.71 %	0	0 %	0.320353	88.42 %
t-SNE & DBSCAN	59.94259	122.29 %	1.09237	100.05 %	0	0 %	0.35958	158.54 %

Table 22: Cluster scorings from running clustering on City Hot Path dataset without derived data. The difference is compared to the scores in Table 18.

Method	Calinski & Harabasz	Diff.	Davies-Bouldin	Diff.	Dunn	Diff.	Silhouette	Diff.
k-means	440.69200	52.82 %	0.85892	170.09 %	0.03406	86.42 %	0.47093	55.63 %
DBSCAN	28.96980	25.47 %	0.54180	201.74 %	0.23784	140.55 %	0.45395	64.95 %
t-SNE & k-means	151.09089	143.68 %	1.78830	155.66 %	0.01275	3036.94 %	0.20416	186.36 %
t-SNE & DBSCAN	405.42261	73.42 %	0.74813	107.39 %	0.07839	1177.79 %	0.43428	91.27 %

Table 23: Cluster scorings from running clustering on Motorway Hot Path dataset without derived data. The difference is compared to the scores in Table 19.

References

[1] Rakesh Agrawal and Ramakrishnan Srikant. "Fast Algorithms for Mining Association Rules". In: *Proceedings of the 20th VLDB Conference* (1994), pp. 487–499. URL: https://www.it.uu.se/edu/course/homepage/infoutv/ht08/vldb94_rj.pdf.

[2] University of Alaska Fairbanks. *Centripetal Acceleration*. http://ffden-2.phys.uaf.edu/211_fall2002.web.dir/shawna_sastamoinen/centripetal.htm. Accessed: 2017-05-18.

[3] Michael Breuß, Laurent Hoeltgen, Ali Sharifi Boroujerdi, and Ashkan Mansouri Yarahmadi. "Highly Robust Clustering of GPS Driver Data for Energy Efficient Driving Style Modelling". In: *arXiv preprint arXiv:1610.02815* (2016).

[4] T. Caliński and J Harabasz. "A dendrite method for cluster analysis". In: *Communications in Statistics* 3.1 (1974), pp. 1–27. DOI: 10.1080/03610927408827101. URL: http://www.tandfonline.com/doi/abs/10.1080/03610927408827101.

[5] G. Castignani, T. Derrmann, R. Frank, and T. Engel. "Driver Behavior Profiling Using Smartphones: A Low-Cost Platform for Driver Monitoring". In: *IEEE Intelligent Transportation Systems Magazine* 7.1 (2015-Spring), pp. 91–102. ISSN: 1939-1390. DOI: 10.1109/MITS.2014.2328673.

[6] Chire. *DBSCAN Illustration*. https://commons.wikimedia.org/wiki/File:DBSCAN-Illustration.svg. Accessed: 2017-05-30.

[7] *Clustering - scikit-learn*. http://scikit-learn.org/stable/modules/clustering.html. Accessed: 2017-06-05.

[8] Zoran Constantinescu, Cristian Marinoiu, and Monica Vladoiu. "Driving Style Analysis Using Data Mining Techniques". In: *Int. J. of Computers, Communications and Control* 5.5 (2010), pp. 654–663. ISSN: 1841-9836. URL: https://www.researchgate.net/publication/228654786_Driving_Style_Analysis_Using_Data_Mining_Techniques.

[9] D. L. Davies and D. W. Bouldin. "A Cluster Separation Measure". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2 (1979-04), pp. 224–227. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1979.4766909.

[10] J. C. Dunn†. "Well-Separated Clusters and Optimal Fuzzy Partitions". In: *Journal of Cybernetics* 4.1 (1974), pp. 95–104. DOI: 10.1080/01969727408546059. URL: http://dx.doi.org/10.1080/01969727408546059.

[11] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: AAAI Press, 1996, pp. 226–231.

[12] Leonard Evans and Paul Wasielewski. "Do accident-involved drivers exhibit riskier everyday driving behavior?" In: *Accident Analysis and Prevention* 14.1 (1982), pp. 57–64. ISSN: 0001-4575. DOI: http://dx.doi.org/10.1016/0001-4575(82)90007-0. URL: http://www.sciencedirect.com/science/article/pii/0001457582900070.

[13] Jiawei Han and Micheline Kamber. *Data Mining Concepts and Techniques*. 2nd. Diane Cerra, 2006, pp. 71–72.

[14] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. *A Practical Guide to Support Vector Classification*. Tech. rep. Department of Computer Science, National Taiwan University, 2003. URL: http://www.csie.ntu.edu.tw/~cjlin/papers.html.

[15] Co-op Insurance. *Your online dashboard for Young Driver Insurance*. http://www.co-opinsurance.co.uk/youngdriverinsurance/your-online-dashboard. Accessed: 2017-05-15.

[16] insurethebox. *Telematics Explained - What is Telematics?* https://www.insurethebox.com/telematics. Accessed: 2017-05-15.

[17] *ITS Platform*. https://www.its-platform.eu/. Accessed: 2017-03-03.

[18] Rizwana Kalsoom and Zahid Halim. "Clustering the driving features based on data streams". In: *Multi Topic Conference (INMIC), 2013 16th International*. IEEE. 2013, pp. 89–94. URL: http://ieeexplore.ieee.org/abstract/document/6731330/.

[19] Joachim Klokkervoll, Samuel Nygaard, Mike Pedersen, Lynge Poulsgaard, Philip Sørensen, and Henrik Ullerichs. "Mining Driving Preferences and Efficient Personalized Routing". In: (2017). URL: http://ppsa.dk/download/uni/p9.pdf.

[20] T. Kohonen. "The self-organizing map". In: *Proceedings of the IEEE* 78.9 (1990-09), pp. 1464–1480. ISSN: 0018-9219. DOI: 10.1109/5.58325.

[21] Teuvo Kohonen. "Self-organized formation of topologically correct feature maps". In: *Biological Cybernetics* 43.1 (1982), pp. 59–69. ISSN: 1432-0770. DOI: 10.1007/BF00337288. URL: http://dx.doi.org/10.1007/BF00337288.

[22] Benjamin Krogh, Nikos Pelekis, Yannis Theodoridis, and Kristian Torp. "Path-based Queries on Trajectory Data". In: *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. SIGSPATIAL '14. Dallas, Texas: ACM, 2014, pp. 341–350. ISBN: 978-1-4503-3131-9. DOI: 10.1145/2666310.2666413. URL: http://doi.acm.org/10.1145/2666310.2666413.

[23] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *Ann. Math. Statist.* 22.1 (1951-03), pp. 79–86. DOI: 10.1214/aoms/1177729694. URL: http://dx.doi.org/10.1214/aoms/1177729694.

[24] Xiaolei Li, Jiawei Han, Jae-Gil Lee, and Hector Gonzalez. "Traffic density-based discovery of hot routes in road networks". In: *International Symposium on Spatial and Temporal Databases*. Springer. 2007, pp. 441–459. URL: http://hanj.cs.illinois.edu/pdf/sstd07_xli.pdf.

[25] Ingenie Services Limited. *Black Box Car Insurance for Young Drivers*. https://www.ingenie.com/. Accessed: 2017-03-06.

[26] Wuman Luo, Haoyu Tan, Lei Chen, and Lionel M. Ni. "Finding Time Period-based Most Frequent Path in Big Trajectory Data". In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. SIGMOD '13. New York, New York, USA: ACM, 2013, pp. 713–724. ISBN: 978-1-4503-2037-5. DOI: 10.1145/2463676.2465287. URL: http://doi.acm.org/10.1145/2463676.2465287.

[27] Laurens van der Maaten and Geoffrey Hinton. "Visualizing High-Dimensional Data using t-SNE". In: 2008, pp. 2579–2605. URL: http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

[28] J. MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297. URL: http://projecteuclid.org/euclid.bsmsp/1200512992.

[29] Y. L. Murphey, R. Milton, and L. Kiliaris. "Driver's style classification using jerk analysis". In: *2009 IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems*. 2009-03, pp. 23–28. DOI: 10.1109/CIVVS.2009.4938719.

[30] Sebastian Raschka. "About feature scaling and normalization". In: *Sebastian Racha. Disques, nd Web. Dec* (2014). URL: http://sebastianraschka.com/Articles/2014_about_feature_scaling.html.

[31] Peter J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427. DOI: http://dx.doi.org/10.1016/0377-0427(87)90125-7. URL: http://www.sciencedirect.com/science/article/pii/0377042787901257.

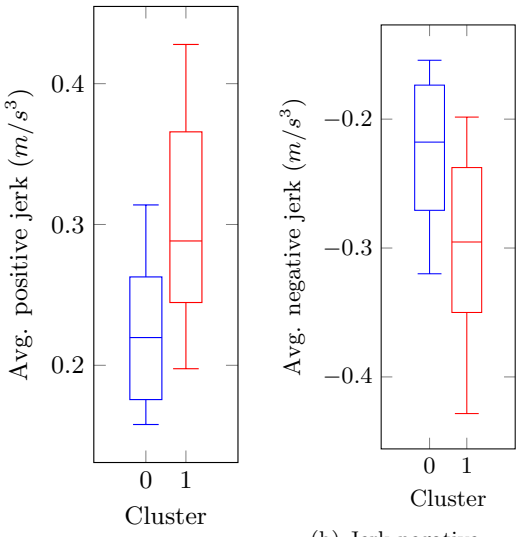
[32] *scikit-learn: machine learning in Python*. http://scikit-learn.org/stable/index.html. Accessed: 2017-05-24.

[33] Telematics. *What Is Telematics and Telematics Technology*. https://www.fleetmatics.com/what-is-telematics. Accessed: 2017-05-15.

[34] Wenshuo Wang and Junqiang Xi. "A rapid pattern-recognition method for driving styles using clustering-based support vector machines". In: *American Control Conference (ACC), 2016*. IEEE. 2016, pp. 5270–5275.

[35] Hong Wei, Yin Wang, George Forman, Yanmin Zhu, and Haibing Guan. "Fast Viterbi map matching with tunable weight functions". In: *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM. 2012, pp. 613–616. URL: https://www.cs.umd.edu/~hyw/papers/gis.pdf.

A. CLUSTERING BOX PLOTS



(a) Jerk positive.

(b) Jerk negative.

Figure 17: City Hot Path jerk using k-means with t-SNE.

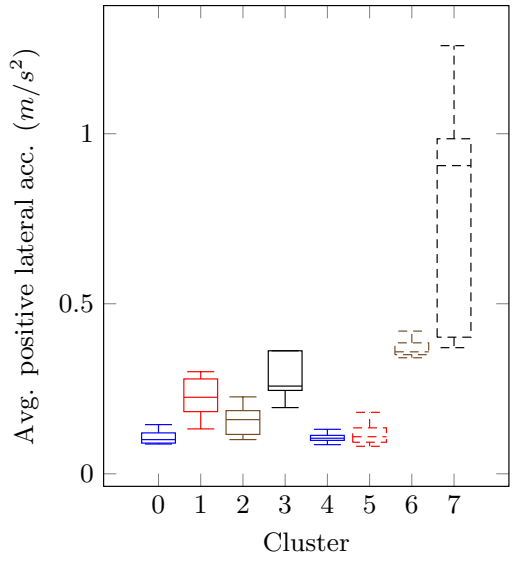
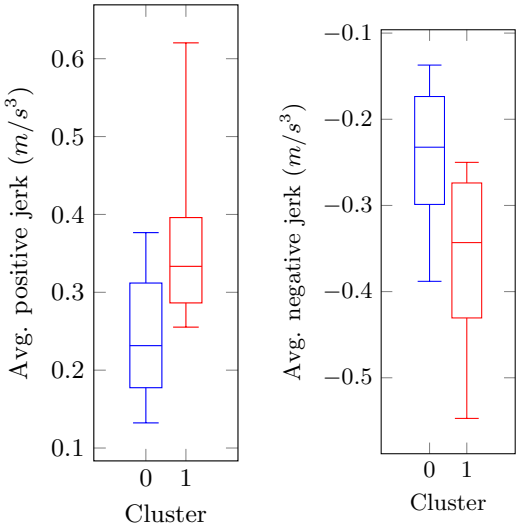


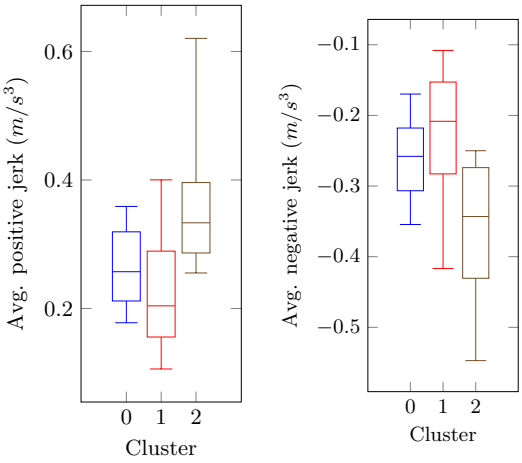
Figure 18: City Hot Path lateral acceleration using DBSCAN with t-SNE.



(a) Jerk positive.

(b) Jerk negative.

Figure 19: Motorway Hot Path jerk using DBSCAN.



(a) Jerk positive.

(b) Jerk negative.

Figure 20: Motorway Hot Path jerk using DBSCAN with t-SNE.