UNIVERSITEIT UTRECHT

Predicting train journeys from smart card data: a real-world application of the sequence prediction problem

Author:
Jelte HOEKSTRA

Advisors:
dr. Ad FEELDERS
prof. dr. Marc VAN KREVELD

Abstract

This study aims to predict the next journey of travelers by train based on smart card data. After preprocessing raw data into features describing journeys, the problem is framed as a sequence prediction instance. Domain modelling issues such as the choice of alphabet, representation of time and the definition of a sequence are discussed. A base alphabet is constructed, and closed frequent pattern mining is proposed as a method of algorithmically extending it. The resulting data encodings are tested against a range of established sequence prediction algorithms. Results show the All-Kth-Order-Markov algorithm outperforms other algorithms by a margin. With regard to pattern encoding, the results are somewhat inconclusive.

January 21, 2016

My love for pattern is so great,
my heart melts for it 'til the dusk of day.
The night knows when it's away,
learn, encode 'til day's dawn.

Its beauty is great,
Wondering mind 'til it sees,
predicting is all I do,
While waiting for the moment, for it to say "I do."

UNKNOWN ALGORITHM

Contents

1 Introduction

1.1 Congestion in the Netherlands

Everyday an average of seven percent of the Dutch population travels at least one journey using public transport [1]. Many of these trips have a commuting purpose and take place within a relatively small timeframe called rush hour. In the Netherlands traffic congestion during rush hours is a serious problem. The overwhelming demand for rush hour trips leads to a shortage of seats and delays due to slow passenger boarding. Typical rush hours are the morning and evening commute on weekdays. Besides these weekday peaks, the Netherlands knows a student peak during Sunday evening when students return from spending the weekend at their parents' place.

1.2 Common solutions

In general, solutions to congestion either decrease traffic demand or increase the capacity of the problematic transport mode. The latter could involve infrastructural improvements such as designated buslanes. Another possibility concerns optimizing existing schedules and removing inefficiencies. Simply adding trips to the schedule or increasing the capacities of transport vehicles, is also a straightforward solution. However, expanding capacity is usually costly and its scope limited: adding more than one buslane has little effect and only so many train carriages fit on a platform. Besides, the extra capacity is only needed for a few brief intervals every week while the extra costs are usually constant. Flexibility in supply is usually difficult to achieve.

Transportation demand management on the other hand, holds more potential for resolving public transport congestion. This may be accomplished

through a wide range of strategies. Simply raising the ticket price or introducing a congestion wage is not desirable in this case. For most people daily travel is mandatory and thus demand will shift to another transport mode. No transport operator will look upon such measures favourably because they compete for the same user base. Also, road traffic congestion is a related problem that is likely to intensify as a result. Another consideration is that affordable public transport is valuable from a sociopolitical standpoint.

Solutions that preserve an operator's user base are likely to meet far less resistance politically. Apart from switching to a different mode of transport, travelers may also respond to congestion wage by traveling at off-rush hours. This way rush hour traffic is dispersed over a larger timeframe, distributing peak travelers over multiple trips. Also, the opposite approach to congestion wage - rewarding people for travelling at off-rush hours - is preferable because it involves a bonus rather than a penalty. Operators do not want to punish users for using their product.

1.3 MyOV

MyOV is a pilot project in the North of the Netherlands (mainly Friesland and Groningen). It attempts to solve the rush hour problem for train routes by stimulating peak travelers to change their schedule. Stimulation comes in the shape of requests for behaviour change, which are made through a mobile application, text message or e-mail. Travelers who respond positively are rewarded with peak-points that can be redeemed against benefits such as travel discounts or a cup of coffee. Besides collecting points, travelers can benefit from an improved travel experience: less crowdedness and a better chance of finding a seated place. A less obvious but possibly quite significant reward is a sense of contribution and collaboration: by changing their schedule, users helped solv-

ing a major problem.

Despite such benefits not all travelers are willing to change their behaviour. Many people are expected at their job at fixed hours. Other people have little need for said rewards, or find changing their behaviour stressful. Yet some others are open to the idea in general, but can participate only under certain circumstances. The latter group is MyOV's primary focus.

All in all a large number of variables factor into such a decision, and the weight of each of these likely varies by person. The MyOV project aims to assemble a strategy to consistently move part of the peak travelers to a different timeslot. This could potentially be achieved by e.g. identifying a group of people with a high compliance rate, or by finding successful combinations of rewards.

Little is known with regard to such combinations, and any presumptions will have to be validated on the fly. As there are no fixed users and the number of users is limited, they should be treated with care. If the project runs out of users, there is no way to learn an effective strategy. Also, with a seeming lack of interest, the project might be terminated prematurely. Thus it is important that users derive benefit from their use: requests should present an opportunity. Most people have a limited patience with mistakes and do not like to be spammed. Even the most forgiving users will only tolerate so many off-target requests before opting out. Thus it is essential to send requests only when necessary.

Change is only desirable if the user travels during the rush hour and on a trajectory that is known to be problematic. Reasonable requests address these circumstances and suggest a specific change. Another obvious requirement is that requests must reach the user before the actual behaviour takes place, i.e. within the window of influence in which the user makes up his mind about

the next trip. If the request comes in too early, it will seem out of place or the user will forget about it. If it comes too late, travel plans will already have been made. This interval could range from hours to days before the actual journey - but probably not weeks. Thus knowing the day on which the journey will take place is a necessary aspect of the prediction.

In summary, the prediction should include

- the day on which the journey will take place
- if the journey will take place during rush hour
- if the journey will be on a problem trajectory

1.4 Predicting the next journey

One straightforward way to obtain travel plans is to simply contact the user and inquire about his plans, e.g. at the start of each week. This seems a bad idea for multiple reasons. First of all, submitting plans would require a significant effort on the part of the user. Apart from posing an inconvenience, self-report is notoriously unreliable, especially given the potential for gaming the system. Finally the user might also be unwilling to submit his plans for privacy reasons.

Another approach is to infer the user's plans from his application use. The MyOV mobile application contains a journey planner which requires users to enter the origin, destination, and date and time of their journey. From these all of the required information can be derived. Although viable, this approach suffers some shortcomings. First of all, use of the planner is optional. Users might have a different preferred planner, or not use a planner at all. For such users there is no basis from which to infer the next journey. Also, users might use the planner only some of the time, hence the acquired plans could be incomplete. Another problem is that the user's planning behaviour might not represent his

actual behaviour. The user might be planning a journey for a friend or relative. Finally, when the user finds out about a connection between his planning behaviour and requests, he might consider this an invasion of his privacy or attempt to game the system, reverse engineering the MyOV strategy.

An alternative approach and the aim of this study, is to predict the user's next journey from his past behaviour. It is assumed that a considerable share of most user's journeys is of structural nature, i.e. these journeys take place according to some schedule or temporal cycle. Prime examples are the 9-to-5 commuter, high-school students and the weekly homebound student. With sufficient examples of previous cycles, the next cycle might be extrapolated.

This study first preprocesses raw travel data into feature vectors representing journeys. These features capture relevant aspects of journeys and are used to define journey categories. Next, the problem is framed as an instance of the sequence prediction problem and several prediction algorithms are introduced. The various modeling concerns encountered when specifying the instance are discussed. Especially the categorization of journeys is of interest. A base alphabet for encoding journeys is presented, and pattern mining is proposed as an approach for algorithmically extending it with subcategories. Finally, the base- and extended alphabet are used to encode the MyOV data and compared across aforementioned prediction algorithms.

2 Preprocessing

2.1 The dataset

Data was collected by scraping the OV-chipkaart logs of MyOV users [1]. It consists of users' public transport trips by train, where a trip describes a ride on one vehicle. The trip starts when the user enters the vehicle and ends when he leaves it. Even if the users forgets to check out, or transfers to a different vehicle without checking out, the trip ends when he leaves the vehicle. Although within the scope of the MyOV project, other modalities such as bus trips are not part of this study. Besides the check-in and check-out times, vehicle departure and arrival times are known. Also, origin and destination stations are included. The dataset contains 40,117 of such trips collectively traveled by 2483 users during the period 01-06-2015 to 01-10-2015.

Consecutive trips make up a journey when they actually belong together: many journeys include transfers between trips. Two trips are considered part of the same journey when the destination of the first trip matches the origin of the second trip. Additionally, the check-in time of the second trip must follow the check-out time of the first trip within at most 35 minutes - the official transfer time according to NS railways [12].

2.2 Features

Features are variables computed from the original data to model the domain in terms of the desired outcome. Although there are few set guidelines for making features, they should help discern between journeys on relevant problem aspects such as peak and off-peak, problem or ordinary trajectory, and notions of

[1] This is part of the MyOV Terms of Service.

date and time (see subsection 1.4). For each of these this study proposes several variants.

2.2.1 Peak

The simplest feature encodes each journey as peak or off-peak. According to NS railways, peak periods are weekdays between 6.30 - 9.00 or 16.00 - 18.30 [11]. An alternative feature further refines this definition by specifying the type of peak: morning peak, afternoon peak, or student peak, where student peak concerns evening travel on Friday or Sunday between 19.00 - 22.00 [2].

2.2.2 Trajectory

Similar to the peak features, a binary feature indicates whether a problem trajectory is involved. Again, a categorical feature specifies the actual trajectory. A list of problematic trajectories was supplied by MyOV (see Appendix A).

2.2.3 Problem

The binary *peak?* and *problem-trajectory* features can be collapsed into one feature, *problem?*, which indicates whether the journey is a problem. This is the logical AND of the peak and trajectory feature.

2.2.4 Date and time

With regard to date and time, a binary feature indicates whether the journey started on a weekday. A categorical feature specifies the day itself. Another feature encodes the hour of the day in which the checkin of the journey took

[2] If some trip within the journey started in these intervals, the journey was marked as a peak journey.

place. Finally, one feature discretizes the time between checkin and departure into short (<5 min.), mean (5-10 min.) and long (10+ min.) categories. A long waiting time could indicate that the user came early to ensure he would not miss the journey. It could also mean that the traveler simply was not in a hurry and could afford to be inefficient. There are many explanations possible here, each of which has different implications with respect to potential compliance. Regardless, the waiting time seems potentially relevant.

2.3 Journey as a feature vector

After features have been computed, the preprocessing is concluded. The resulting data is a set of journeys, c.q. feature vectors (see Figure 1). The dataset contains 37.064 of such journeys. Hence by far most journeys do not contain transfers between trains.

Feature	Value
Peak?	True
Problem-trajectory?	True
Problem?	True
Weekday	Monday
Peak	Morning
Problem-trajectory	Leeuwarden-Groningen
Weekday?	True
Hour	8
Waiting-time	Short

Figure 1: An example of a journey feature vector. Features that are part of the solution requirements are displayed in gray.

3 Sequence prediction

3.1 Problem definition

Sequence prediction concerns predicting the next event of a sequence of events from a finite alphabet Z. In this case, the alphabet is a set of categories to which journeys might belong. It maps journey feature vectors to symbols denoting the categories. Prediction algorithms construct a model from a set of Z encoded training instances that are considered a representative sample. This model can be used to predict the next event of a new sequence. When making a prediction, the model receives a prefix subsequence as input. These are the observed events up to moment of prediction. The algorithm then outputs a prediction of the symbol that will follow the prefix.

Some authors refer to this problem as discrete sequence prediction and contrast it with the related problems of sequence extrapolation and time-series prediction [9] [8]. In the former problem, the focus is on finding an underlying rule or formula that sufficiently explains the data. A solution is a stream of predictions, rather than one prediction. Given the complexity of the problem, it seems unlikely that a rule is found to produce such a stream. It is not really a problem if the solution is blackbox either, hence knowing the underlying rule is not that beneficial. Possibly the sequence extrapolation framework might be applied to more specific predictions, i.e. to predict a subset of behaviour.

Time-series prediction is more concerned with regression problems in which the predictions concerns the value of a numerical variable as a function of time. The case at hand requires the prediction of discrete categories. Hence, discrete sequence prediction (sequence prediction from here on) seems a better fit. Although the problem at hand could probably be framed as these related problems, this is outside the scope of this study.

Sequence prediction is a general problem with a wide range of use cases. Some clever applications are adaptive file compression and content preloading [13]. In adaptive file compression, prediction is used to encode the most probable remaining characters with the fewest bits. Content preloading is about anticipating a person's next decision to preload e.g. website content, thus reducing perceived latency.

A sequence prediction setup involves two aspects: the domain model and the prediction algorithm. The former concerns which events are in the alphabet, the way time is represented in the model and the definition of a sequence.

The choice of algorithm involves decisions about whether the Markov assumption holds, whether the training data contains noise and whether the algorithm should abstain from prediction with little information.

3.2 Algorithms

Because of its wide applicability, the sequence prediction problem has been tackled from a considerable number of angles. Besides adaptations of general supervised learning techniques like neural networks [18], some dedicated algorithms exist.

Dedicated solvers can be roughly divided into two groups: probability-based solvers that extract transition probabilities between subsequences and comparison-based solvers that compare input sequences against a condensed representation of the training sequences.

Probability-based solvers assume the Markov property, namely that future states (predictions) depend only upon a fixed number of past states. This number is called the order of the model and indicates its complexity. Markov models capture states of the world and transition probabilities between such states. These probabilities are determined as observed from training data. Conse-

quently, any transition not occurring in the training data cannot be handled by the model. Thus an extensive training set is required. Even then, noisy instances remain problematic.

3.2.1 Probability-based

The first algorithm to apply the probability-based approach to sequence prediction was Prediction by Partial Matching (PPM) [2]. It represented subsequences of events as states and the probability of subsequent occurrence as transitions. Both of these are stored in a frequency table. During prediction, the most likely transition from the prefix available in the table is predicted.

PPM gave rise to a number of successors, the first of which was the Transition Directed Acyclic Graph (TDAG) [7] that builds a Markov tree where paths represent observed prefix subsequences. Even simpler is the Dependency Graph algorithm (DG) [13], essentially a first-order Markov model in that predictions are based solely on the latest observed event. A few years later Pitkow and Pirolli experimented with higher order models and found a trade-off between accuracy and applicability: higher order models produce more accurate predictions but are less likely to produce a prediction at all. They match longer prefix subsequences, i.e. it is less likely that the prefix has already been observed. Hence they suffer from aforementioned data sparsity issues. Hinging of this insight, they devised All-Kth-order-Markov (AKOM) [15], an algorithm that selects the highest order prediction as learned from the training data.

3.2.2 Comparison-based

Much more recently a different approach was proposed with the Compact Prediction Tree (CPT) algorithm [6]. Different from the Markovian algorithms discussed so far, CPT compares the prefix subsequence at hand against a lossless

but compressed tree structure of all training sequences. It predicts the overall most frequently occurring suffix event among similar sequences. In the absence of similar sequences - possibly due to noise - prefix events are recursively discounted until sufficient matches have been found. Although CPT seemingly achieves higher accuracies than most probability-based algorithms [6], it entails a higher spatial complexity and higher prediction time. Two years later the same authors aimed to improve these aspects with further optimizations in CPT+ [5].

4 Domain model

Whereas the sequence prediction problem concerns sequences of events, the available data concerns a database of feature vectors, c.q. journeys and their characteristics. The domain model bridges this chasm by dividing journeys into disjoint categories (events) and by defining the limits of serial association (sequences). As time plays an important role in the prediction, its representation is also a major modelling decision. Time can be modelled implicitly using an event's relative position in the sequence, or explicitly as part of the definition of an event.

4.1 Implicit vs explicit time

In an implicit model sequences have fixed length where each index represents a time interval. For example, if the sequence represents a week, then the sequence could contain 7 events, one for each day of the week. This approach works well as long as exactly one journey takes place each day. An extra event is needed to represent the case that no journey takes place on a day (see Figure 2). An implicit model predicts what type of journey takes place in the next time

interval. Although subtly different from predicting the next journey, predicting the next interval seems permissible in the case at hand. The benefit of implicit modeling is that the alphabet is considerably simplified: the absence of time in the event definition makes that fewer events are needed to describe all kinds of journeys.

A	B	A	X	X	A	X

Figure 2: Cells represent the 7 days of the week. X represents the case that no journey took place on a day.

A challenge using the implicit approach is handling multiple journeys within the same interval, which is a problem that actually arises in the dataset (see Figure 3. The straightforward solution is to extend the alphabet to include multiple journey events. However the resulting combinatorial explosion makes this infeasible.

Another solution is to shrink time intervals such that each journey again has its own interval. Before long however, this will result in a race to the bottom, requiring ever smaller intervals, where the interval size is essentially the lowest common denominator of inter-event times. Many of these new intervals will be empty (see Figure 4), and as most sequence prediction algorithms look back only so many events, soon enough the noise from empty intervals will drown out the actual journeys.

An alternative solution is to select and encode only one of the journeys. If pessimistic predictions are preferred, the most problematic journey might be selected. However this strategy selectively leaves out some of the data, which might harm performance in nefarious ways: the missing journeys might play a significant role in characterizing the sequence. Another problem occurs when the model is used to predict more than one journey on the same day. The train-

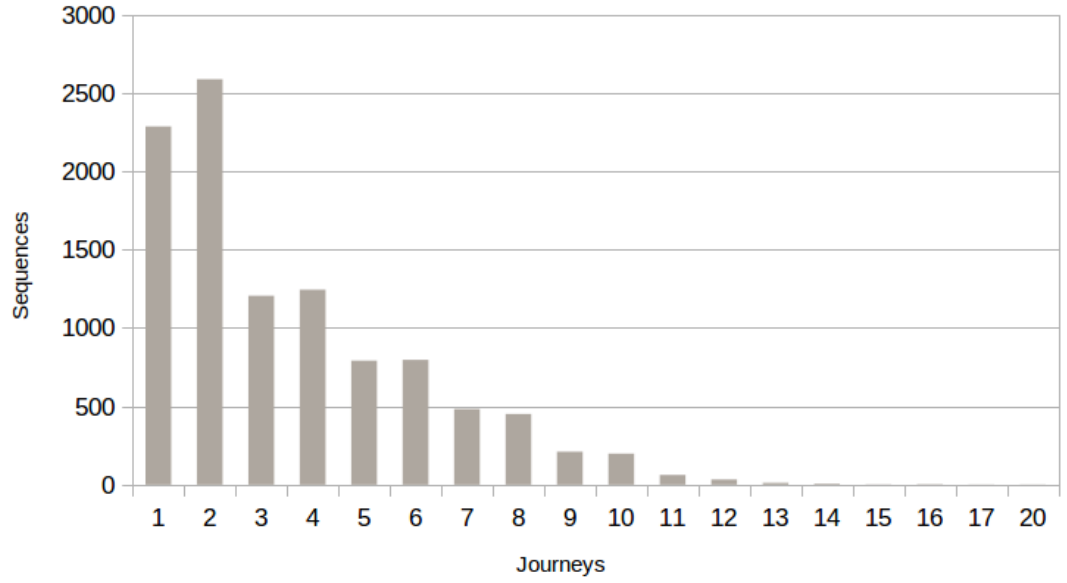

Figure 3: Histogram of sequence lengths. Quite a few sequences have more than 7 journeys, which means that multiple journeys occur on one day.

A	X	X	X	X	B	X	A	X	A	X	X	X	X

Figure 4: Using more but smaller intervals. This results in many empty intervals (X).

ing data does not contain examples of days with multiple journeys, hence the prediction will always predict for some day in the future - never for the same day. Also if more than one problematic journey occurs, alphabet extension or interval shrinkage might still be necessary.

All in all, an implicit model has the benefit of a smaller alphabet but entails some problems. An explicit model on the other hand, incorporates a notion of time in the alphabet, i.e. the definition of an event includes temporal features. For example the day of the week and week number would be selected as temporal features. From these the date could be deduced but this might result in

a significant increase of the alphabet size. To limit the number of variants, it helps to still bound the sequence temporally by e.g. requiring that the sequence contains events only from some specific week. If the sequence spans this week, then its events need only include the day of the week to deduce the date.

Explicit modeling avoids many of the pitfalls of implicit modeling at the cost of a larger alphabet. In the case at hand, this seems a worthwhile trade. Letting sequences span a week and including the day of the week, the alphabet becomes 7 times as large. With the minimally descriptive *problem?* feature (see subsubsection 2.2.3), the resulting alphabet has just 14 events [3]. The dataset shows that each of these events still covers a considerable number of journeys (see Table 1).

Problem?	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Yes	881	832	771	584	528	0	63
No	6446	6331	6149	5096	5046	2352	1985

Table 1: The number of journeys for each of the 14 events. Journeys on Saturday are never a problem by definition.

4.2 Sequences

Starting out with a series of journeys for each user, the question is if and how to divide these into sequences. The straightforward approach is to let a series be a sequence, i.e. have one sequence represent the travel history of one user. In order to predict the date, this would require an even larger alphabet expansion than the aforementioned weekly representation (see subsection 4.1. Also, some

[3] Actually, there are only 13 events because journeys on Saturday are never a problem due to the definition of the *peak?* feature (see subsubsection 2.2.1).

sequence prediction algorithms require the training data to be split into multiple training sequences [5].

Thus data has to be divided into sequences. The mechanism of division - temporal or every so many events - is implied by the model's framing of time (see subsection 4.1). This leaves questions of how many. If sequences represent real time intervals, then how large should the intervals be? If a sequence has a fixed number of events, how many? Besides arguments involving the alphabet size, it is worth noting that sequences need to have a minimum length. Without a minimum length, there would be no prefix subsequence.

Domain specific cycles also factor into the decision. If sequences coincide with natural patterns, it seems more reasonable to consider these to be independent samples. This is an underlying assumption of most model building procedures.

With sequences representing travel weeks, most of these requirements are satisfied. It is the shortest temporal structure with sufficient journeys to support prediction. For most people, this Monday is probably more like the next Monday than any other day of the week. The same cannot be said of the first day of a fortnight or the first day of the month. This representation would result in a total of 10,368 sequences.

4.3 Events

In defining events, the first concern is to ensure that predicted events convey all necessary information as required by the solution requirements (see subsection 1.4). Thus the prediction includes whether the journey is a problem and the date on which it occurs.

The next step is to expand the alphabet beyond the minimum by refining these base events into subevents, introducing subtler distinctions. If an event is

characterized by a combination of feature values, then the remaining features could be used for refinement. Let some event be the combination of a problem journey on a Friday, then the *hour* feature might be used to further divide journeys into subevents for timeslots. Consider an alphabet $Z = \{A, B\}$ and its refinement $Z' = \{A_1, A_2, B\}$, where A_1 and A_2 represent two types of A event. In principle, the refined alphabet Z' allows for more complex learning and rule discovery (see Figure 5). Thus expanding the alphabet potentially leads to improved prediction. However, with too refined an alphabet data becomes sparse. In the extreme case, each journey would have its own category. Another problem is that very specific alphabets encourage algorithms to overfit: the algorithm learns a large number of infrequent sequential associations.

$$A \to B \qquad\qquad A_1 \to B$$
$$A \to A \qquad\qquad A_2 \to A_1$$

Figure 5: Two sequence databases with alphabet $Z = \{A, B\}$ (left) and alphabet $Z' = \{A_1, A_2, B\}$ (right). Encoding with Z' enables the extraction of rules of a higher prediction certainty. However these rules might be overfitting.

Thus refinement of an event comes at a cost, and events should be refined only if this increases the algorithm's ability to differentiate between sequences (see Figure 5). This decision not only depends on the journeys, but also on their order of occurrence and division into sequences. Even then, there is a trade-off between the benefits of improved discernment and the costs of alphabet expansion with respect to prediction performance. For this reason finding an optimal alphabet is a complex undertaking, and the alphabet is usually found empirically.

An alternative approach is to automatically generate an alphabet from the set of journeys. This is essentially the task of finding a strict partitioning clus-

tering. However some essential structure is already dictated by the solution requirements (see subsection 1.4), thus the actual question is whether it would help to refine these categories even more, i.e. to cluster within these given categories. Clustering algorithms often have difficulty deciding on the number of clusters, and for the same reason such algorithms will have a hard time deciding whether a clustering is appropriate at all [10]. Yet a clustering should be used only if the data in a given category exhibits interesting patterns, hence using a clustering algorithm out of the box might not be the best approach. Instead, this study proposes a solution involving pattern mining, where interesting patterns make up subcategories, c.q. subevents [4].

5 Subevents from patterns

5.1 Closed frequent pattern mining

Pattern mining is a fundamental problem in data mining that concerns finding interesting associations in a transaction database, i.e. a database that holds a set of binary attribute vectors. Common applications are market basket analysis, protein sequencing and fraud detection systems [19]. With some exceptions [4], interesting patterns are frequent patterns, c.q. patterns that occur often in the database. Frequency might be defined by a minimum support threshold, or with respect to the support of other patterns.

Although a very debatable standpoint, in the case at hand, frequent subevents are arguably more valuable than infrequent subevents. First of all, the alphabet contains at least 14 events, hence considerable refinement is already in place.

[4] An additional motivation is that this study originally concerned pattern mining of individual travelers. Later on the goal was changed to predicting the next journey, while a pattern mining setup had already been engineered.

Secondly, given the preference for a smaller alphabet, if a subevent occurred very infrequently, it might be better not to include it. The gain from the smaller alphabet might beat the lack of refinement. Thus infrequent subevents are unlikely to be helpful. Finally, frequent subevents could be considered undiscovered elements of the domain, especially when they are specific on many features. An experiment is required to find out whether these intuitions make any sense.

Frequent pattern mining often suffers from the so called pattern explosion, i.e. there are too many frequent patterns to evaluate. Many of these patterns are actually subpatterns, i.e. they cover instances all of which also belong to some stronger, more restrictive pattern. As the stronger pattern conveys more information, it is usually more interesting to discover. Patterns that are not the subpattern of any other pattern are called closed [14]. They cannot be strengthened without lowering the support. For example if all journeys on Friday took place during rush hours, then the subpattern 'journeys on Friday' is not closed. Closed patterns constitute a lossless summary of the discovered patterns.

5.2 Encoding conflicts

Although using closed patterns reduces the possibility, encoding conflicts may arise when a journey falls in multiple categories. There are two ways in which a journey might be covered by more than one frequent closed patterns. The first is when two hierarchically related patterns are both frequent. Let 'problem journeys on Monday' and 'problem journeys on Monday at 11am' be two closed frequent patterns, i.e. they do not have the same support but their support is higher than the minimum support threshold. Then both patterns are valid subevents, and e.g. a problem journey on Monday at 11am belongs to both categories (see Figure 6). Which of the subevents should be used to encode

the journey? The more specific subevent is actually a subevent of the other subevent, i.e. a subsubevent. Similarly subevents are more specific than events, in which case the journey is encoded as the subevent. Thus the more specific (subsub)event should be used to encode the journey. This also makes intuitive sense because the more specific pattern is a closer match.

Feature	Value
Weekday	Monday
Problem?	True

Feature	Value
Weekday	Monday
Problem?	True
Hour	11

Feature	Value
Weekday	Monday
Problem?	True
Hour	11

Figure 6: Two closed frequent patterns (left) and a (truncated) journey (right). The patterns are hierarchical. The journey belongs to both patterns.

The second conflict concerns the case when two subevents are on an equal footing. This can happen when the subevents use different features in their definition (see Figure 7). There seems no straightforward way to resolve this conflict. Adding combination events for pattern combinations is not an option due to resulting alphabet explosion. Thus a selection rule should be used, several of which will be experimented with. Candidate selection rules are the most frequent subevent, the least frequent subevent, and random selection.

Finally, journeys not covered by any subevent are simply encoded as their respective base events.

Feature	Value
Weekday	Monday
Problem?	True
Waiting-time	Short

Feature	Value
Weekday	Monday
Problem?	True
Hour	9

Feature	Value
Weekday	Monday
Problem?	True
Waiting-time	Short
Hour	9

Figure 7: Two closed frequent patterns (left) and a (truncated) journey (right). The patterns are not hierarchical. The journey belongs to both patterns.

6 Experimental design

Before summarizing the experimental setup, the type (individual or population) of model, the importance of the week definition and the division of data into training and testing sequences will be discussed.

6.1 Individual vs population model

Travel behaviour can be described on both a population and an individual level. On the population level, there is behaviour shared among groups of travelers. For example typical 9-to-5 workers are expected to have a somewhat similar travel schedule. Models trained on the population data are less likely to overfit on the noise of a few incidental journeys or to be thrown off by some traveler's week of homestaying due to flu. Cold start problems on new users are also less of an issue.

Individual models on the other hand, adapt to the peculiarities of specific travelers. With sufficient data, these models can deliver personalized predictions which are potentially more accurate.

Both individualized and population approaches have their merits, and as they model different but both important components of travel behaviour, an

ensemble model might prove effective. Such a model could use predictions from individual and population models as weighted input for a final verdict. Over time, as more data is being gathered, the weight is likely to shift from the population prediction to the individualized prediction.

At present, the MyOV dataset has a limited amount of data per user (see Figure 8. Therefore, experimentation involves only population models. This means that data of individual users is grouped together in collective train and test sets.

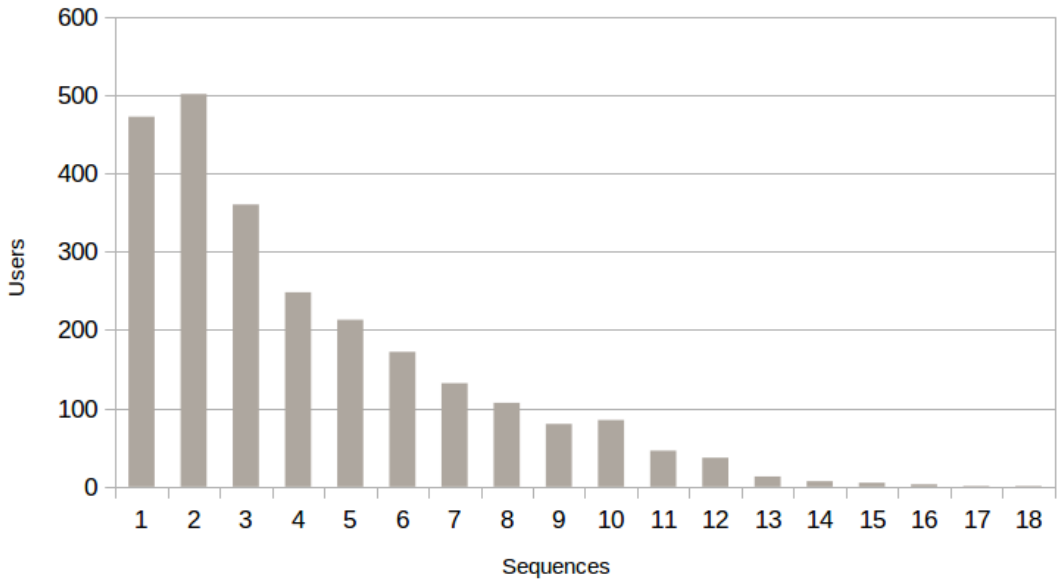

Figure 8: Histogram of the number of sequences per user.

6.2 Definition of the week

Given the division of data into travel weeks (see subsection 4.2), the question is on which day we cut off one week and start the next. This matters because days at the start of the week are predicted with less information, i.e. a shorter prefix

subsequence, than days near the back. If we decide that weeks start on Monday, then there will always be fewer journeys in this week to prime a prediction for Wednesday, than for example Sunday. Thus predictions are better equipped to predict days that happen to be at the back of our week, i.e. the days before the cut-off day. Therefore it seems that the cut-off day is an important parameter.

For this reason 7 experiments have been conducted, one for each possible definition of a week [5]. That is, for each experiment the day on which the week starts (and ends) shifts by one day (see Figure 9).

Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue
A	A	B			A	A	B	A	B

Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue
A	A	B			A	A	B	A	B

Figure 9: The week before (above) and after (below) shifting by one day.

6.3 Train and test set

The next important aspect is the separation of available data into a train and test set. Each user should contribute the same number of weeks to the training and testing data to ensure that the peculiarities, e.g. excessive travel, of any particular traveler will not unduly influence the model. Also, data from the same period is used for each traveler. This makes it easier to control for global effects such as fewer journeys in a holiday week.

[5] Because full weeks were used, these experiments cover slightly different data. Hence technically they are incomparable, as any difference could be attributed to the small difference in data.

To minimize the chance that a user's travel behaviour changed between the training and testing weeks, these weeks must be consecutive. Having assumed that a user's travel behaviour does not change within this period, arguably there is no reason testing must take place in the last week. Predicting past sequences with a model trained on future data seems legitimate given the assumption that no significant behaviour change took place within the data gathering period: The weeks are samples from the same distribution. Allowing backwards prediction would also facilitate cross-validation which might be beneficial given the limited amount of data.

However, in reality there will always be some linear dependence between consecutive travel weeks of a user. By allowing the week definition to shift, this study already acknowledges such. Backwards prediction could affect the experiment in unforeseen ways. Thus this study takes a number of consecutive travel weeks and separates these into a consecutive block of training weeks and a consecutive block of test weeks.

The data collection period is from 01-06-2015 to 01-10-2015, i.e. 122 days or about 17 weeks [6]. It is common to have a train set at least 3 times larger than the test set. Following this observation, the train set should cover at least 3 weeks. With 3 training weeks and 1 test week, we are looking for a block of 4 consecutive weeks.

The number of users with at least 3 journeys [7] in each consecutive block was computed for all blocks. Analysis shows the dataset contains only one such block with at least 100 users (see Table 2). Several 3-week blocks with higher numbers were available, but the 4-week block yields more sequences overall (see Appendix H). Thus data from the period 2015-07-27 to 2015-08-29 was

[6] For some week definitions there are 16 weeks because weeks with dates that fall outside the data collection period, i.e. weeks at the start or end, were removed.

[7] Less than 3 journeys results in very short prefix subsequences.

used. The first 3 weeks of each user serve as training data for the model. The last week of each user serves as testing data. This results in approximately 450 train weeks and 150 test weeks.

Cut-off day	Common users
Monday	147
Tuesday	160
Wednesday	176
Thursday	175
Friday	140
Saturday	140
Sunday	137

Table 2: Common users per starting day for the selected period 2015-07-27 to 2015-08-29

6.4 Evaluation

The data is preprocessed and encoded with a custom Python script. The patterns were mined using the AprioriClose algorithm [14] [8]. This and the sequence prediction algorithms have been implemented in SPMF [3], a Java data mining library used in this study. It comes with an evaluation framework, which was adapted to compensate for the larger alphabet size of the pattern encoding. Predictions were deemed correct as long as the predicted subevent belonged to the correct base event. This is fair because the relevant part of the prediction - *problem?* and *weekday* - is predicted correctly in such cases.

[8] Nominal features are temporally transformed to binary features because the pattern mining algorithm requires a transaction database.

The SPMF evaluation framework divides test sequences into three parts (see Figure 10): the context, the prefix and the suffix. The prefix is used as input to algorithms and the suffix is the part that is to be predicted. The context subsequence is discarded. The prefix subsequence length is set to 10, the largest prefix possible given the computational resources [9]. This means that the input subsequence contains at most 10 journeys. The suffix length is set to 1 to ensure that the predicted event is indeed the next event rather than one of the next events.

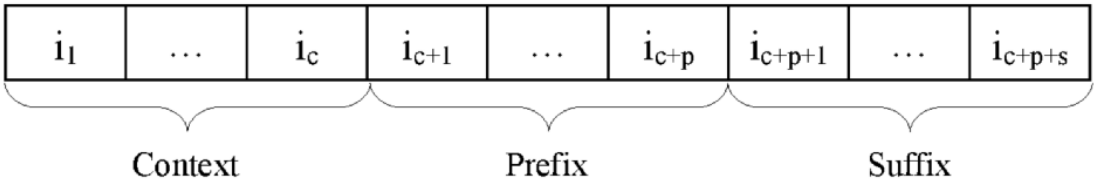

Figure 10: Test sequences are divided into three parts: the context, the prefix and the suffix [5].

To clarify, this means that prefix subsequences include all but the last journey of the week. The distribution of this last journey varies with the cut-off day, which lends extra weight to the hypothesis that the cut-off day is a significant parameter. If the week is defined such that the last journey strongly varies, this should affect the prediction difficulty.

As discussed before, some algorithms cannot handle sequences which were not learned from the train data. This might happen when a sequence occurs in the test set but not the train set, or if parameter settings restrict the algorithm from learning additional complexity. Whenever this happened, the algorithm would simply fail to predict [10]. Thus predictions could result in a success, a

[9]Note that some algorithms used only part of the prefix in accordance with their order parameter.

[10]This might be useful if actual predictions turn out to be more accurate than those of a similar

failure and no match. The overall accuracy is defined as the ratio of successful predictions to the number of testing instances (including no match).

6.5 Parameters

Each of the aforementioned algorithms is included in SPMF. In addition to these, a baseline predictor was added. This predictor simply predicts the event that occurred most as the last event of the week in the training data. With regard to algorithm parameters, Gueniche et al. already reported parameters that performed well across a wide range of datasets [6] [5], and for the most part this study adopted their recommendations (see Appendix B). Most of these parameters affect only the training phase of the respective algorithm. Yet some represent support thresholds which are needed during prediction time. Please consult the original papers for a detailed description.

7 Results and analysis

Algorithms were trained on the train set. The resulting models were used to predict both on the train set itself (Appendix C) and the test set (Appendix D). The resulting accuracies differ significantly as can be glanced from (Appendix E) (Appendix F).

As a first observation, the baseline predictor scores remarkably different on the train and test sets. This indicates they have a somewhat different distribution. If the difference is due to variance, it will diminish with a larger dataset. The other explanation is that the train and test set are samples from a different distribution. As train and test weeks represent different periods, a change in-between these periods might explain the digression.

predictor that guesses.

One such change might be the end of the summer holidays. Primary and high school started on the 16th of August, which falls in the third train week [16]. Also from this date onwards, many students regained the right to travel by public transport free of charge [17]. Thus, increased student travel might have affected the experiment.

7.1 Algorithms

With regard to algorithmic performance, the first observation is that AKOM appears to score best across the board. It was outperformed on just 3 experiments and only by a small margin. Compared to most other algorithms, AKOM is far less sensitive to the cut-off day. This is important because a production setup could be required to predict on any day, i.e. take advantage of a long prefix subsequence by choosing the appropriate cut-off day.

The extremely high accuracies on the train set, especially compared to the test accuracies, suggest that AKOM overfits the data. With respect to the algorithm, the difference between base and pattern encoding is like a complexity parameter: pattern encoded data enables more complex models. For some cut-off days AKOM scores significantly higher when a base encoding is used. This supports the notion that AKOM overfits because performance declines with increased model complexity.

If AKOM overfits, performance is expected to go up with more data. Whereas base encoded data still outperforms pattern encoded data in this experiment, with more data pattern encoded data may give AKOM a significant boost. This is quite promising given the relatively small dataset.

Most other algorithms perform less convincing. PPM's scores follow a trend similar to AKOM but slightly inferior. This is reassuring as AKOM is essentially a powered-up version of PPM: Whereas PPM only has a first order model,

AKOM has access to higher order predictions. The privilege appears to pay off.

Besides PPM, DG is strongly affected by the cut-off day. Nevertheless, its best scores are impressive. Perhaps DG could be used as a predictor on some cut-off days. TDAG scores extremely low, especially on pattern encoded data. Further inspection reveals this is due to passivity rather than wrong predictions. TDAG fails to predict on most instances. The prime reasons seems that the test set contains unseen data but this is not a sufficient explanation: Even on the train set TDAG fails to predict about half of the cases (see Appendix G). Increasing the tree height parameter might resolve this issue but requires an excessive amount of space. To a lesser degree CPT and CPT+ also suffer from passivity. In the case of CPT, the problem persists on the train set, similar to TDAG. CPT+ on the other hand recovers completely which is promising: More data will probably diminish its passivity.

7.2 Cut-off day

For most algorithms the cut-off day has a significant effect on performance. Across the board cut-off day Saturday or Sunday yields lower accuracies than e.g. Wednesday or Thursday. The dichotomy roughly corresponds to the difference between weekdays and weekend. These are expected to be different because the weekend does not include commuting travel.

An explanation for the relative performance is that the week contains more weekdays than weekend, hence behaviour learned from weekdays will tend to dominate the model. Also, the dataset contains disproportionally more journeys on weekdays than on weekend days which exacerbates the effect.

As an exception, AKOM is little affected by the cut-off day parameter. This suggests that other algorithms might perform more reliably by manually tuning the model complexity to the cut-off day.

7.3 Pattern encoding

The results are somewhat inconclusive with respect to pattern encoding the data. This is especially true where it matters, i.e. at the high end of the algorithm spectrum. Among the three variants of pattern encoding no significant winner comes forth, although using random patterns is often slightly inferior. This suggests that frequency as a selection criterion matters because inconsistent, random selection (slightly) decreases accuracy. One explanation for the small differences between the three variants is that for many journeys only one possible pattern remained after resolving encoding conflicts. Thus no selection was required, and the three methods encoded the journey with the same pattern.

Compared to base encoding, the train accuracies suggest pattern encoding has greater potential but this is not reflected in the test accuracies. Perhaps more data or different features will make a difference.

8 Discussion

The outcomes of this study should be considered signposts for future research rather than definitive results. The experiment was conducted with little data. The data belongs to MyOV users that traveled at least 3 times every week in August 2015, i.e. a select group of travelers and time period. Train and test data are quite likely not fully representative.

Besides collecting more data, a larger dataset could be instantly obtained by dropping one of the constraints of this study (subsection 6.3). Perhaps not all training weeks need to be collected from the same consecutive period because population-wide effects are expected to cancel each other out. Perhaps the data collection period does not need to be consecutive because travel behaviour is

expected to be more stable over time.

Besides the small dataset, the reader should note that results are more ordinal than numerical: Comparing between base and pattern encoded data could be considered model tuning, in which case an additional test set, i.e. validation set, is required to determine the absolute accuracies of the selected model. Additional parameter tuning, especially of the model order, is likely to improve performance of most algorithms given the effectiveness of AKOM.

Before conducting research beyond this study, it seems advisable to validate its main results as outlined below.

1. Construct a larger dataset, either by collecting more data or by weakening the data selection constraints. Select a period without potential biases.

2. Divide the dataset into a train, validation, and test set. The validation and test should have a similar size. The train set should be at least 3 times larger than the validation and test set.

3. Repeat the experiment in this study. Tune the algorithm parameters to the validation set. Select the most effective algorithms and test these on the test set.

A possible bias of the current experimental design is that only the last journey of testing weeks are predicted, which ensures relatively long prefix subsequences on average. In practice, journeys before (what later turns out to be) the last journey of the week will also have to be predicted. With respect to some fixed week definition, on average these predictions will have shorter prefix subsequences, which might affect prediction accuracy. Nevertheless, in these cases a model trained on a more suitable week definition might be used. Ideally, a model is selected with a week definition such that the prefix subsequence is long enough but not too long: The week should end a few days after the last

registered journeys to make sure that the next journey still takes place within the same week. This is not a clear cut definition, hence results obtained in this study might be somewhat optimistic: Obtaining the same results under similar circumstances in a practical setting might be harder to achieve.

With regard to limitations, this study implicitly assumes that predictions are performed immediately after the last journey was registered, i.e after the moment that the last journey of the prefix subsequence took place. This because the absence of a journey is not registered, nor are categories corresponding to past days excluded from the prediction space. To illustrate this argument, let the week be defined such that it starts on Thursday. If the last journey of a prefix subsequence took place on Monday and the moment of prediction is Monday, predictions should only predict categories on Monday, Tuesday, or Wednesday. However if the moment of prediction is Tuesday, only Tuesday and Wednesday categories would make sense as predictions. The experimental setup does allow Monday categories to be predicted, hence it implicitly assumes the prediction was performed immediately after the journey took place [11].

Future research might want to explore mechanisms for taking into account the absence of journeys and performing delayed predictions. One solution would be to exclude categories of past days from the prediction space. However this would require modification of the prediction algorithms. A modelling solution is to introduce an event for the case that no journey took place on a day. To avoid the issues associated with implicit modeling of time, a distinct event might be introduced for each day of the week. Thus in the resulting sequences each day is represented as a no-journey event, or as one or multiple

[11] Technically it would also allow predictions on categories before Monday but this is different because sequences that violate the order of days in the week cannot occur in the training set. Hence predictions that violate the order are unlikely to happen in practice.

journey events. If a delayed prediction is desired, an absence of journeys could be modelled by appending the respective no-journey events to the end of the sequence before prediction. This approach would of course require that training sequences also include these new events.

Including such no-journey events would have more benefits. In this experiment algorithms predict the category of the last journey of the week. Thus it is given that another journey will take place in the remaining days of the week. This simplifies the real problem because it is also possible that no journey took place at all. Inclusion of no-journey events would resolve this limitation but result in mostly predictions of one day ahead: If no journey takes place tomorrow than the respective no-journey event is expected as a prediction. However the next journey can still be predicted by appending the predicted no-journey event to the end of the sequence. This updated sequence can again be used as input to predict the next event. This process may be repeated until another journey event, i.e. the actual next journey, comes up. At first glance this seems like building a house of cards because predictions from previous iterations are used as input to the next. Yet this is not so different from the current setup in which the no-journey events are implied by the absence of journey events.

With regard to pattern encoding data, it is recommended to experiment with additional features. Weather circumstances and traffic jams for example could be of interest. Before definitively adding new features to the feature pool, it seems advisable to first extend the base alphabet with the new feature. If the resulting encoding improves accuracy, then this is reason to believe the feature could indeed be a relevant component of subcategories. In addition to such practice runs, it could help to start with a clear dichotomy, e.g. rain vs no rain, or frost vs no frost. Additional granularity could be added incrementally until no further improvements are found.

With more data it becomes interesting to assemble specialized models. Examples would be separate models for weekdays and weekend, and discerning between ordinary days and special days like commemorative days or the holiday period. Such days are outliers, likely to skew the model.

Clustering user groups might also prove beneficial. With this strategy cold start problems are avoided while the modelled behaviour is still more relevant than behaviour learned from the entire population. For long term users individual models could also be constructed. Finally, it would be interesting to see if combining these model in an ensemble method increases accuracy.

9 References

[1] CBS. Public transport statistics. http://statline.cbs.nl/Statweb/publication/?VW=T&DM=SLNL&PA=81125NED&D1=a&D2=0&D3=0&D4=0-10,23-41,52-54&D5=a&HD=151112-1536&HDR=G1,G2,T,G4&STB=G3, 2015. Accessed: 2015-11-12.

[2] CLEARY, J. G., AND WITTEN, I. Data compression using adaptive coding and partial string matching. *Communications, IEEE Transactions on 32*, 4 (Apr 1984), 396–402.

[3] FOURNIER-VIGER, P., GOMARIZ, A., GUENICHE, T., SOLTANI, A., WU., C., AND TSENG, V. S. SPMF: a Java Open-Source Pattern Mining Library. *Journal of Machine Learning Research (JMLR) 15* (2014), 3389–3393.

[4] GENG, L., AND HAMILTON, H. J. Interestingness measures for data mining: A survey. *ACM Computing Surveys (CSUR) 38*, 3 (2006), 9.

[5] GUENICHE, T., FOURNIER-VIGER, P., RAMAN, R., AND TSENG, V. S. CPT+: Decreasing the time/space complexity of the compact prediction tree. In *Advances in Knowledge Discovery and Data Mining*. Springer, 2015, pp. 625–636.

[6] GUENICHE, T., FOURNIER-VIGER, P., AND TSENG, V. S. Compact prediction tree: a lossless model for accurate sequence prediction. In *Advanced Data Mining and Applications*. Springer, 2013, pp. 177–188.

[7] LAIRD, P., AND SAUL, R. Predictive caching using the TDAG algorithm. *NASA Ames Research Center: Technical Report FIA 92-30* (1992).

[8] LAIRD, P., AND SAUL, R. Discrete sequence prediction and its applications. *Machine learning 15*, 1 (1994), 43–68.

[9] LAIRD, P., SAUL, R., AND DUNNING, P. A model of sequence extrapolation. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory* (New York, NY, USA, 1993), COLT '93, ACM, pp. 84–93.

[10] MILLIGAN, G. W., AND COOPER, M. C. An examination of procedures for determining the number of clusters in a data set. *Psychometrika 50*, 2 (1985), 159–179.

[11] NS. Peak travel. `http://www.ns.nl/reizigers/klantenservice/klantenservice/voorwaarden-en-folders/daluren.html`, 2015. Accessed: 2015-11-12.

[12] NS. Transfer time. `http://www.ns.nl/reizigers/ovchipkaart/reizen/overstappen.html`, 2015. Accessed: 2015-11-12.

[13] PADMANABHAN, V. N., AND MOGUL, J. C. Using predictive prefetching to improve world wide web latency. *SIGCOMM Comput. Commun. Rev. 26*, 3 (July 1996), 22–36.

[14] PASQUIER, N., BASTIDE, Y., TAOUIL, R., AND LAKHAL, L. Discovering frequent closed itemsets for association rules. In *Database Theory—ICDT'99*. Springer, 1999, pp. 398–416.

[15] PITKOW, J., AND PIROLLI, P. Mining longest repeating subsequences to predict world wide web surfing. In *Proc. USENIX Symp. On Internet Technologies and Systems* (1999), p. 1.

[16] SCHOOLVAKANTIES-NEDERLAND.NL. Schoolvakanties 2015. `http://www.schoolvakanties-nederland.nl/zomervakantie-2015.html`, 2015. Accessed: 2015-12-17.

[17] STUDENTENREISPRODUCT.NL. Studentenreisproduct geldigheid zomerperiode. http://www.studentenreisproduct.nl/vragen/geldigheid-studentenreisproduct/, 2015. Accessed: 2015-12-17.

[18] SUN, R., AND GILES, C. L. Sequence learning: from recognition and prediction to sequential decision making. *IEEE Intelligent Systems 16*, 4 (2001), 67–70.

[19] ZAKI, MOHAMMED, J., AND MEIRA JR., W. *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

Appendix A Problem trajectories

From	To
Nieuweschans	Groningen
Veendam	Groningen
Leeuwarden	Groningen
Groningen	Leeuwarden
Delfzijl	Groningen
Roodeschool	Groningen
Emmen	Zwolle
Stavoren	Leeuwarden
Harlingen	Leeuwarden

Table 3: Problem trajectories according to MyOV.

Appendix B Algorithm parameters

Algorithm	Parameter	Value
PPM	order	1
AKOM	order	5
TDAG	maximumDepth	7
TDAG	maxTreeHeight	6
DG	lookAhead	4
CPT	splitLength	10
CPT	maxLevel	99
CPT+	CCFmin	2
CPT+	CCFmax	4
CPT+	CCFminsup	4
CPT+	MBR	2
CPT+	TB	100%

Table 4: Parameter settings of the prediction algorithms.

Appendix C Overall train accuracies

Cut-off day	Encoding	DG	TDAG	CPT+	CPT	PPM	AKOM	BasePred
Monday	base	42.404	29.932	54.422	47.166	61.224	73.243	37.415
Monday	px-mfp	59.184	44.444	55.556	64.626	69.841	97.959	37.415
Monday	px-lfp	60.091	45.351	59.184	66.213	70.295	98.413	37.415
Monday	px-random	58.73	45.351	59.41	66.213	69.161	98.413	37.415
Tuesday	base	61.875	30	62.5	51.042	50.208	73.333	60.208
Tuesday	px-mfp	69.375	45.208	68.75	65.417	68.542	96.667	60.208
Tuesday	px-lfp	67.708	46.667	63.75	66.042	67.708	97.083	60.208
Tuesday	px-random	69.583	45.833	65.208	66.25	67.292	96.667	60.208
Wednesday	base	68.939	36.174	71.78	52.652	69.697	83.144	64.205
Wednesday	px-mfp	76.705	45.265	76.705	64.394	76.515	98.106	64.205
Wednesday	px-lfp	76.515	45.455	76.136	63.826	75.189	98.864	64.205
Wednesday	px-random	75.758	46.023	77.462	64.583	76.136	98.674	64.205
Thursday	base	68	34.667	71.81	53.905	68.952	80.381	64
Thursday	px-mfp	75.429	46.095	76.571	64.571	78.476	98.095	64
Thursday	px-lfp	74.286	45.143	75.619	64.571	78.476	98.095	64
Thursday	px-random	76.762	45.905	75.619	64.381	78.476	98.476	64
Friday	base	66.667	31.19	69.524	56.429	68.095	79.286	65.238
Friday	px-mfp	77.857	43.571	75.714	67.619	78.571	98.095	65.238
Friday	px-lfp	79.524	43.81	77.619	67.619	78.333	97.857	65.238
Friday	px-random	77.857	43.81	77.381	68.095	79.286	98.095	65.238
Saturday	base	64.286	26.429	64.762	54.762	61.905	80.714	60.238
Saturday	px-mfp	75	38.571	74.048	71.19	78.81	98.571	60.238
Saturday	px-lfp	75.238	37.619	74.762	70.714	76.905	98.095	60.238
Saturday	px-random	74.286	36.905	73.333	70.476	77.619	98.571	60.238
Sunday	base	44.282	23.601	54.988	51.825	56.448	78.345	42.336
Sunday	px-mfp	69.586	36.01	59.124	72.749	71.533	98.54	42.336
Sunday	px-lfp	71.533	35.523	58.151	72.019	73.236	98.783	42.336
Sunday	px-random	71.533	36.74	60.341	73.479	72.993	99.027	42.336

Table 5: Overall train accuracies. Each row contains the overall accuracy for one experiment. Encodings are 'base' for base encoding, 'px-mfp' for most frequent pattern encoding, 'px-lfp' for least frequent pattern encoding and 'px-random' for random pattern encoding.

Appendix D Overall test accuracies

Cut-off day	Encoding	DG	TDAG	CPT+	CPT	PPM	AKOM	BasePred
Monday	base	36.054	18.367	48.98	42.177	55.102	63.265	34.014
Monday	px-mfp	50.34	2.721	41.497	34.694	57.823	62.585	34.014
Monday	px-lfp	51.701	4.762	42.857	36.735	60.544	62.585	34.014
Monday	px-random	46.259	2.721	38.095	34.014	61.905	63.946	34.014
Tuesday	base	60	16.25	61.25	53.125	55	63.125	62.5
Tuesday	px-mfp	61.25	2.5	55	46.875	60.625	63.125	62.5
Tuesday	px-lfp	63.75	2.5	51.25	45	60.625	61.875	62.5
Tuesday	px-random	63.125	1.875	48.125	46.875	54.375	57.5	62.5
Wednesday	base	69.886	18.75	67.045	56.818	72.727	74.432	59.091
Wednesday	px-mfp	76.136	7.386	63.636	53.409	72.727	77.841	59.091
Wednesday	px-lfp	76.136	6.25	65.909	52.273	75	77.273	59.091
Wednesday	px-random	75	5.114	61.364	52.841	71.591	76.136	59.091
Thursday	base	66.857	20.571	68	60	68.571	76.571	64
Thursday	px-mfp	69.143	5.143	64	56	70.857	76.571	64
Thursday	px-lfp	69.714	4.571	64.571	56	70.286	76.571	64
Thursday	px-random	68.571	4.571	69.143	53.143	68	73.143	64
Friday	base	32.143	33.571	17.857	13.571	38.571	69.286	19.286
Friday	px-mfp	47.143	4.286	32.857	23.571	57.857	62.857	19.286
Friday	px-lfp	35	4.286	27.857	22.857	44.286	59.286	19.286
Friday	px-random	41.429	2.857	27.857	22.143	53.571	56.429	19.286
Saturday	base	26.429	44.286	13.571	7.143	38.571	65.714	5
Saturday	px-mfp	28.571	9.286	17.857	6.429	54.286	59.286	5
Saturday	px-lfp	32.143	7.857	16.429	6.429	57.143	60	5
Saturday	px-random	27.857	6.429	15.714	8.571	47.857	52.143	5
Sunday	base	27.007	51.095	13.869	7.299	38.686	72.993	4.38
Sunday	px-mfp	25.547	11.679	21.898	8.029	55.474	59.124	4.38
Sunday	px-lfp	27.007	9.489	14.599	8.759	62.044	62.774	4.38
Sunday	px-random	25.547	10.949	14.599	8.029	51.095	56.204	4.38

Table 6: Overall test accuracies. Each row contains the overall accuracy for one experiment. Encodings are 'base' for base encoding, 'px-mfp' for most frequent pattern encoding, 'px-lfp' for least frequent pattern encoding and 'px-random' for random pattern encoding.

Appendix E Overall train accuracies heatmap

	DG	TDAG	CPT+	CPT	Mark1	AKOM	BasePred
Monday	42.404	29.932	54.422	47.166	61.224	73.243	37.415
	59.184	44.444	55.556	64.626	69.841	97.959	37.415
	60.091	45.351	59.184	66.213	70.295	98.413	37.415
	58.73	45.351	59.41	66.213	69.161	98.413	37.415
Tuesday	61.875	30	62.5	51.042	50.208	73.333	60.208
	69.375	45.208	68.75	65.417	68.542	96.667	60.208
	67.708	46.667	63.75	66.042	67.708	97.083	60.208
	69.583	45.833	65.208	66.25	67.292	96.667	60.208
Wednesday	68.939	36.174	71.78	52.652	69.697	83.144	64.205
	76.705	45.265	76.705	64.394	76.515	98.106	64.205
	76.515	45.455	76.136	63.826	75.189	98.864	64.205
	75.758	46.023	77.462	64.583	76.136	98.674	64.205
Thursday	68	34.667	71.81	53.905	68.952	80.381	64
	75.429	46.095	76.571	64.571	78.476	98.095	64
	74.286	45.143	75.619	64.571	78.476	98.095	64
	76.762	45.905	75.619	64.381	78.476	98.476	64
Friday	66.667	31.19	69.524	56.429	68.095	79.286	65.238
	77.857	43.571	75.714	67.619	78.571	98.095	65.238
	79.524	43.81	77.619	67.619	78.333	97.857	65.238
	77.857	43.81	77.381	68.095	79.286	98.095	65.238
Saturday	64.286	26.429	64.762	54.762	61.905	80.714	60.238
	75	38.571	74.048	71.19	78.81	98.571	60.238
	75.238	37.619	74.762	70.714	76.905	98.095	60.238
	74.286	36.905	73.333	70.476	77.619	98.571	60.238
Sunday	44.282	23.601	54.988	51.825	56.448	78.345	42.336
	69.586	36.01	59.124	72.749	71.533	98.54	42.336
	71.533	35.523	58.151	72.019	73.236	98.783	42.336
	71.533	36.74	60.341	73.479	72.993	99.027	42.336

Figure 11: Heatmap of the overall train accuracies (see Appendix C). Darker colours indicate higher accuracies. Encoding labels have been omitted.

Appendix F Overall test accuracies heatmap

	DG	TDAG	CPT+	CPT	Mark1	AKOM	BasePred
Monday	36.054	18.367	48.98	42.177	55.102	63.265	34.014
	50.34	2.721	41.497	34.694	57.823	62.585	34.014
	51.701	4.762	42.857	36.735	60.544	62.585	34.014
	46.259	2.721	38.095	34.014	61.905	63.946	34.014
Tuesday	60	16.25	61.25	53.125	55	63.125	62.5
	61.25	2.5	55	46.875	60.625	63.125	62.5
	63.75	2.5	51.25	45	60.625	61.875	62.5
	63.125	1.875	48.125	46.875	54.375	57.5	62.5
Wednesday	69.886	18.75	67.045	56.818	72.727	74.432	59.091
	76.136	7.386	63.636	53.409	72.727	77.841	59.091
	76.136	6.25	65.909	52.273	75	77.273	59.091
	75	5.114	61.364	52.841	71.591	76.136	59.091
Thursday	66.857	20.571	68	60	68.571	76.571	64
	69.143	5.143	64	56	70.857	76.571	64
	69.714	4.571	64.571	56	70.286	76.571	64
	68.571	4.571	69.143	53.143	68	73.143	64
Friday	32.143	33.571	17.857	13.571	38.571	69.286	19.286
	47.143	4.286	32.857	23.571	57.857	62.857	19.286
	35	4.286	27.857	22.857	44.286	59.286	19.286
	41.429	2.857	27.857	22.143	53.571	56.429	19.286
Saturday	26.429	44.286	13.571	7.143	38.571	65.714	5
	28.571	9.286	17.857	6.429	54.286	59.286	5
	32.143	7.857	16.429	6.429	57.143	60	5
	27.857	6.429	15.714	8.571	47.857	52.143	5
Sunday	27.007	51.095	13.869	7.299	38.686	72.993	4.38
	25.547	11.679	21.898	8.029	55.474	59.124	4.38
	27.007	9.489	14.599	8.759	62.044	62.774	4.38
	25.547	10.949	14.599	8.029	51.095	56.204	4.38

Figure 12: Heatmap of the overall test accuracies (see Appendix D). Darker colours indicate higher accuracies. Encoding labels have been omitted.

Appendix G No match rate

Cut-off day	Encoding	DG	TDAG	CPT+	CPT	PPM	AKOM	BasePred
Monday	base	0	54.649	1.587	31.293	0	0	0
Monday	px-mfp	0	54.649	0.68	31.293	0	0	0
Monday	px-lfp	0	53.741	0.454	31.293	0	0	0
Monday	px-random	0	53.968	0.907	31.293	0	0	0
Tuesday	base	0	52.917	0	31.875	0	0	0
Tuesday	px-mfp	0	53.333	1.25	31.875	0	0	0
Tuesday	px-lfp	0	52.292	1.458	31.875	0	0	0
Tuesday	px-random	0	53.125	1.042	31.875	0	0	0
Wednesday	base	0	53.977	0	34.091	0	0	0
Wednesday	px-mfp	0	53.788	0.758	34.091	0	0	0
Wednesday	px-lfp	0	53.977	1.136	34.091	0	0	0
Wednesday	px-random	0	53.598	0.568	34.091	0	0	0
Thursday	base	0	53.143	0	33.143	0	0	0
Thursday	px-mfp	0	53.143	1.333	33.143	0	0	0
Thursday	px-lfp	0	54.286	0.762	33.143	0	0	0
Thursday	px-random	0	53.714	1.333	33.143	0	0	0
Friday	base	0	55.952	0	31.429	0	0	0
Friday	px-mfp	0	55.714	1.905	31.429	0	0	0
Friday	px-lfp	0	55.952	1.667	31.429	0	0	0
Friday	px-random	0	55.714	1.905	31.429	0	0	0
Saturday	base	0	62.143	0.238	27.143	0	0	0
Saturday	px-mfp	0	60.952	1.429	27.143	0	0	0
Saturday	px-lfp	0	61.667	1.429	27.143	0	0	0
Saturday	px-random	0	62.381	1.667	27.143	0	0	0
Sunday	base	0	64.234	0	24.088	0	0	0
Sunday	px-mfp	0	63.017	1.946	24.088	0	0	0
Sunday	px-lfp	0	63.99	1.217	24.088	0	0	0
Sunday	px-random	0	62.53	1.703	24.088	0	0	0

Table 7: No match rate on train data. Each cell contains the ratio of no match instances to total testing instances. Encodings are 'base' for base encoding, 'px-mfp' for most frequent pattern encoding, 'px-lfp' for least frequent pattern encoding and 'px-random' for random pattern encoding.

Appendix H Travel data availability

First week id	Last week id	Cut-off day	Common_users
3	5	Monday	115
3	5	Tuesday	122
3	5	Wednesday	146
3	5	Thursday	154
3	5	Friday	120
3	5	Saturday	121
3	5	Sunday	113
8	10	Monday	176
8	10	Tuesday	190
8	10	Wednesday	212
8	10	Thursday	210
8	10	Friday	176
8	10	Saturday	189
8	10	Sunday	193
8	11	Monday	147
8	11	Tuesday	160
8	11	Wednesday	176
8	11	Thursday	175
8	11	Friday	140
8	11	Saturday	140
8	11	Sunday	137
9	11	Monday	194
9	11	Tuesday	205
9	11	Wednesday	231
9	11	Thursday	246
9	11	Friday	220
9	11	Saturday	218
9	11	Sunday	213

Table 8: Periods with at least 100 users and sequences of at least 3 journeys. Number of common users on all week definitions.