Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Kaja Sofie Lundgaard

Identify the Most Trafficked Roads Using Sequence Alignment Algorithms

Master's thesis in Informatics
Supervisor: Svein-Erik Bratsberg
May 2022

NTNU
Norwegian University of
Science and Technology

Kaja Sofie Lundgaard

Identify the Most Trafficked Roads Using Sequence Alignment Algorithms

Master's thesis in Informatics
Supervisor: Svein-Erik Bratsberg
May 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

The mapping and processing of traffic data contribute to creating better and more sustainable transportation alternatives. This thesis investigates the use of sequence alignment algorithms for comparing trajectories with the goal of mapping traffic. Trajectory similarity is an extensively researched topic, yet sequence alignment algorithms have not been tested for this purpose before. In this study, three sequence alignment algorithms are implemented to compare trajectories to identify the most trafficked road in a city. The algorithms output overlapping sub-trajectories that are assigned a score based on frequency and length. The result was proved to consist of frequent sub-trajectories that covered the busiest streets in the city of Porto.

Sammendrag

Kartlegging og behandling av trafikkdata bidrar til å skape bedre og mer bærekraftige transportalternativer. Denne oppgaven undersøker bruken av sekvenssammenstillingsalgoritmer for å sammenligne GPS-spor med mål om å kartlegge trafikk. Likhet i GPS-spor er et mye undersøkt emne, men algoritmer for sekvenssammenstilling har ikke blitt testet for dette formålet før. I denne studien er tre sekvenssammenstillingsalgoritmer implementert for å sammenligne GPS-spor for å identifisere den mest trafikkerte veien i en by. Algoritmene sender ut overlappende del-spor som tildeles en poengsum basert på hyppighet og lengde. Resultatet ble bevist å bestå av hyppige del-spor som dekket de travleste gatene i byen Porto.

Acknowledgement

I want to thank my supervisor Svein-Erik Bratsberg for his support and guidance. He has provided me with weekly advice, motivation, and feedback that has been an immense asset to completing this thesis.

I would also like to thank my friends and family for their support and comfort as the deadline approached. A special thank you to August Sollesnes Solvang for listening to my complaints, in addition to providing advice, support, and proofreading skills.

Table of Contents

1. Introduction

1.1. Motivation

Smart cities are growing worldwide [35], along with the need for sustainable transportation [14]. According to the European Commission, a smart city is a place where digital solutions contribute to the efficiency of traditional networks and services to benefit its inhabitants and businesses [15]. Transportation accounts for 25% of the EU's greenhouse gas emissions, so the need for greener and cleaner transportation alternatives is growing [14]. 70% of the world's population will reside in urban areas by 2050, according to United Nations [52]. Europes population is already at 75% [53]. This growth will contribute to new challenges like increased congestion, traffic accidents, pollution, and waste. Smart Cities are an initiative to help solve new challenges emerging as cities grow [6]. The amount of digital tools like mobile phones, GPS, and RFID [58] increases, making movement and transportation data more available than ever. Smart cities use this transportation data for city planning purposes and, in particular, to create environment-friendly solutions [35]. Previously, analyses of cities have been based on individuals. However, with the rise of computing power and data availability, many large IT companies look more widely at applications based on groups [8]. Data-driven decision-making will contribute to making the city intelligent and automated [42]. The method introduced in this thesis can be used for comparing trajectories to identify people's transportation habits. This can help change bus routes to fit people's travel behavior and adapt traffic lighting to congestion formation [45, 7]. Cycling fields can be built where they are most needed, and the most pressured roads can be prioritized for renovation. This field can also help people choose their route based on traffic congestion or detecting incidents and anomalies [27, 34, 23]. Ridesharing, or carpooling, is a climate change measure this method can be used for [13]. Instead of driving alone to work every day, people can team up with others driving the same distance at the same time.

1.1.1. Trajectory Similarity and Sequence Alignment

Trajectory similarity can be used to map traffic and is central to solving transportation-related challenges. It is a widely explored field and includes calculating the similarity between different types of trajectories. Trajectories are a general term for an object's moving path in space. The space can include road networks used by cars or open space used by boats or planes [43, 44]. The data could be spatial or spatio-temporal and real-time or historical. Trajectory similarity can include movements, shapes, and time for comparison. This thesis explores historical traffic data running on roads in a city, analyzed in euclidean space. Shape or time will not be included in this analysis, only sub-trajectories overlapping in space.

In this thesis, sequence alignment algorithms will be used to solve the problem of trajectory similarity. Sequence alignment has the goal of finding the optimal alignment between two sequences. The term originates from bioinformatics, where sequence alignment is used to identify similarities between DNA or protein sequences [59]. The optimal alignment is found by computing the similarity or distance between the sequences. That is done by comparing every element from the first sequence to every element from the second sequence. Then, the best combination of elements achieving the highest score will be considered the optimal alignment.

Calculating the distance between two trajectories is much more complex than calculating the distance between two points. Trajectory distance consists of perpendicular distance, parallel distance and angle distance [28]. However, when using a sequence alignment algorithm, the sequence is split up into the elements it consists of and instead uses these to discover the distance, or rather the similarity. The sequence alignment algorithms will not measure the distance between the trajectories but their similarity value. This value is based on the distances between the trajectories' points. The sequence alignment algorithms simplify the problem, and

we only have to deal with the single distance between two points. Even speed, time, acceleration, and vectors are insignificant for this problem and method.

1.2. Goals and Research Questions

The project aims to investigate the performance of sequence alignment algorithms for trajectory similarity and use this information to identify the most trafficked roads in a city.

Three sequence alignment algorithms will calculate the similarity between trajectories and identify where two trajectories overlap. The most represented sub-trajectories will be gathered in a result list, sorted on length and frequency. The results will be investigated, and the algorithms' traits explored to determine if they can be used for this purpose and find out which performed better. Needleman-Wunsch is a global alignment algorithm that originates from bioinformatics and is made for comparing amino acid sequences of proteins [32]. Smith-Waterman is made for the same purpose but is a local alignment algorithm [47]. Dynamic Time Warping is very similar to Needleman-Wunsch and is also a global alignment algorithm, but it originates from comparing time series and speech recognition [9, 40].

The research questions for this study are the following:

Research question 1 Which of the three algorithms Needleman-Wunsch, Smith-Waterman, and Dynamic Time Warping is most suited for identifying overlapping sub-trajectories?

Research question 2 What are the differences and similarities between these algorithms for identifying the most frequent sub-trajectories in a dataset?

1.3. Research Method

The algorithms will be implemented and adapted to use GPS trajectories as input. They will compare the trajectories and output the sub-trajectories where they overlap. These sub-trajectories will be gathered and rated based on length and frequency. The longest and most frequent sub-trajectories will be ordered in a list based on these two numbers. The three complete result lists of the algorithms will be compared. They shall be tested on a dataset of movement data to investigate if they perform satisfactorily. They will also be evaluated on their ability to discover matching trajectories and what traits they have concerning how these are discovered.

1.4. Terminology

This section will describe a set of key terms used in this thesis.

A **trajectory** is defined as "a sequence of time-stamped locations" [16] and is, in this thesis, mainly used related to the field of trajectory similarity. A **polyline** represents, in this study, the list of coordinate points. A **track** consists of a polyline and an id and can be divided into sub-tracks. **Sequences** and **subsequences** are more closely related to bioinformatics and alignment algorithms. A sequence is a general term for a series of elements in a list. A subsequence is a subset of a sequence.

A **result track** is a track that made it into the result list. The **result list** consists of several result tracks and their score. There are three result lists, one for each of the three algorithms.

Throughout the thesis, some abbreviations will be used for the names of the algorithms. Needleman-Wunsch will be referred to as NW, Smith-Waterman as SW, and Dynamic Time Warping as DTW.

1.5. Thesis Outline

Section 1: Introduction

This chapter introduces the project's motivation, goal, and research questions. It also includes a short explanation of the research method, terminology, and this thesis outline.

Section 2: Background Theory

The background theory explains the theoretical background of traffic mapping and sequence alignment. It also explains the related work already done in this field.

Section 3: Methods

The method chapter explains how the three sequence alignment algorithms work and how they are adapted to the problem of trajectory similarity.

Section 4: Implementation and Experiments

How the algorithms are implemented is described in this chapter. It includes information about the dataset and how the data is preprocessed. This chapter also introduces the evaluation metrics and the experiments to be conducted.

Section 5: Results

The Results chapter displays how the algorithms performed in the experiments.

Section 6: Discussion

In this chapter, the algorithms are discussed in light of the results. The reason behind the results will be delved into, and the algorithms' strengths and weaknesses will emerge.

Section 7: Conclusion and Further Work

The last chapter summarizes the thesis and discusses the contributions and limitations of the work.

2. Background Theory

2.1. Traffic Mapping

There are many purposes for traffic mapping and, therefore, many methods to obtain the information wanted. Studies are exploring air and noise pollution [10, 29, 56] and mapping traffic load for bridges to know when to repair [60]. Some use real-time speed and acceleration data to identify where there might be congestion or an incident [38, 23]. Google Maps is based on this data and has a feature where it identifies and reports real-time traffic incidents. They do this with the help of every mobile phone on the road that is using the Google Maps app [49]. The app helps them track the speed and acceleration of cars to identify where there is congestion. That is a standard method to detect congestion [57]. Because Google Maps is the most popular traffic map app, it is also the most accurate. Some incidents are also based on user reports. Exactly how they are processing the data is not known.

Identifying the most trafficked road in a city is not a widely explored field. There is much research done on identifying data traffic in digital networks [36, 21]. It does not differ much from roads in a city, so these methods can provide insight into how these problems are solved today. In 2007, Hyytïa and Virtamo studied traffic load detection in a wireless multihop network [21]. The project's goal was to solve the problem of uneven load. Some connections in the network experienced congestion while others were unused. They invented a way of detecting traffic load on connections to change routing. The traffic load was decreased by 40% in comparison to shortest-path routing. It was done by computing a lower bound for how large the traffic load could be. It is also a real-time problem and not a historical data analysis problem. If the traffic load exceeds the lower bound, the traffic is rerouted through another path.

Traffic congestion and flow can be interpreted as a traffic network where nodes are intersections and edges are the roads between intersections [37]. Traffic in a city can, thus, be mapped as a weighted flow graph. This method can help map the traffic to detect congestion areas and measure the flow. In addition to identifying the traffic in the city, this also identifies how much traffic the roads can handle. A common preprocessing step to identify which trajectories overlap is called map-matching. Map-matching is a method where trajectories are matched to a map to identify which road the trajectory is traveling [39]. The map used is a predefined road network, similar to a graph of nodes and edges. The method requires a similarity algorithm to match the trajectory to the right road. Dynamic Time Warping has been used for this purpose [55].

Jiwon Kim and Hani S. Mahmassani wrote a paper about trajectory clustering in a traffic network [26]. They skipped the map-matching step and compared the trajectories directly with each other using the Longest Common Subsequence algorithm. LCSS is a sequence alignment algorithm, like the algorithms used in this thesis, and is widely used for trajectory similarity [51]. Their goal was to identify patterns in the traffic network by finding representative dense subsequences from the clusters created with the LCSS algorithm. The goal of this thesis is similar, except that the clustering step is skipped and other sequence alignment algorithms are used. Kim and Mahassani expect two trajectories' level of overlap to measure their similarity. In addition to the timestamp for the trajectory, this number creates the measurement behind the clustering. They obtained great results and identified several clusters representing the patterns of trajectories.

2.2. Sequence Alignment

Sequence alignment is a term originating from bioinformatics, and the algorithms are made with DNA and protein sequences in mind [59]. The main difference between comparing DNA sequences and GPS trajectories is that human DNA sequences have an expected similarity of 99.4 % [50], while GPS trajectories might not match at all. DNA and protein sequences consist of only four different letters. They can either match or not match. GPS trajectories can consist of an infinite number of coordinate points. The probability of a match is much lower. When computing the similarity, too much time will be spent comparing points in sequences far away from each other.

Another difference is that two coordinate points do not have to be precisely the same to match. They only have to be closer than a given threshold. This means that two coordinate points might not match, but they are both considered a match to a third point lying between them. The algorithms were not made to handle this situation, and their precision might be affected.

A trajectory is a term that can include many different methods and types of data. Trajectories concerning traffic in a city often use a road network. Hwang et al. have conducted several studies on trajectory similarity [19, 20] with a specific focus on road networks. The alignment methods are based on objects in euclidean space and not on road networks. Therefore, the computation will be performed differently. The studies conducted in road networks compute the distance between points along a road. Alignment algorithms compute the distance in a straight line. Even though this study is based on road networks and trajectories following the road network, using a straight line to compute distance is not unsuitable. The distances above a given threshold will not be considered either way, and if there are two trajectories on parallel neighboring roads, they will be above this threshold. However, following a road network will improve the accuracy, but it comes with a higher computation cost.

Sequence alignment has been used to compare trajectories before. Usually, global alignment algorithms are used to measure the similarity between two trajectories. A study conducted by Abraham and Lal in 2012 used a global sequence alignment algorithm to compute the spatio-temporal similarity of moving vehicles in a road network [4]. They used a global alignment matrix to compute a similarity value between two trajectories to measure how similar they were. The problem was closely related to the studies conducted by Hwang, but Abraham and Lal used sequence alignment to solve it. They achieved a better result and claimed this was due to the reduced complexity of the binary-encoded scheme used in the sequence alignment. In 2002, Vlachos used the Longest Common Subsequence method to discover similarities between trajectories [54]. The method of computing the longest common subsequence between two sequences is very similar to the local alignment algorithm Smith-Waterman. Their method worked very well, especially for trajectories with much noise, which is typical for the LCSS algorithm. Their LCSS-inspired algorithm did perform better than DTW. DTW is not as robust to noise and did not cluster as accurately as their algorithm. Nevertheless, it did not perform poorly.

Both Chen et al. and Toohey and Duckham have conducted studies on Dynamic Time Warping on user trajectories with GPS tracks [11, 51]. They conclude that DTW is too sensitive to noise to be a good enough algorithm for user-generated GPS trajectories. That is because GPS is not accurately representing a vessel's movement. GPS will generate noise points. How much it will affect the result of the different algorithms is fascinating to explore. How influential noise is, depends on the study and the problem to be solved.

Using DNA sequence alignment algorithms to compare GPS trajectories is a very limited explored area. In recent times, some studies have been conducted. Needleman-Wunsch was used for comparison and clustering purposes in 2021 [62]. They obtained good results for classification and found that mismatches were not necessary to achieve these good results. However, Needleman-Wunsch has been applied

to numerous other new fields like comparing navigation paths on the World Wide Web [17].

Smith-Waterman is the algorithm with the least previous research on trajectories. However, it was used in a research-based on constructing mobility maps via tracking [61]. Smith-Waterman was used as a matching technique to match a tracked path to a target area. The experiment worked nicely, and Smith-Waterman solved the problem well.

Both Smith-Waterman and Needleman-Wunsch have been used to compare students' eye movements while solving problems [25, 12]. The algorithms were used to compare the eye-tracking paths to identify which students answered correctly or incorrectly on the tasks. These algorithms have many different usabilities, and we can assume the algorithms will work to some extent.

3. Methods

The algorithms introduced in this chapter will be used to compare the trajectories. They use a very similar method as they are all sequence alignment algorithms. However, their minor differences could have a considerable impact on the results. They are adapted to fit the data and problem explored in this study, which is different from their original purpose.

Every algorithm will describe how it initially works and what is done to adapt them to this problem. Details of the implementation, dataset, and preprocessing of the data are described in Section 4.

3.1. Needleman-Wunsch

Needleman-Wunsch is a global alignment algorithm comparing amino acid sequences of two proteins [32]. In short, it compares two sequences of letters and returns the optimal global alignment of the two sequences. It also calculates a number that represents the similarity of two sequences.

3.1.1. Implementation

We cannot require a perfect match to show similarity when adapting this algorithm to coordinates rather than letters. The distance between two points is measured and is considered a match if the distance is below a predefined threshold. If the distance is above the threshold, they are considered a mismatch. The threshold is a hyperparameter of this implementation and was set to be 30 meters. That is because there are a maximum of 20 meters between the points from the same trajectory. The reason behind the 20 meters is explained further in Section 4.2 about generating waypoints between existing points. Different thresholds were tested out, and thresholds below 30 did not catch all points from the matching trajectories. Thresholds above 30 found too many matches from parallel roads.

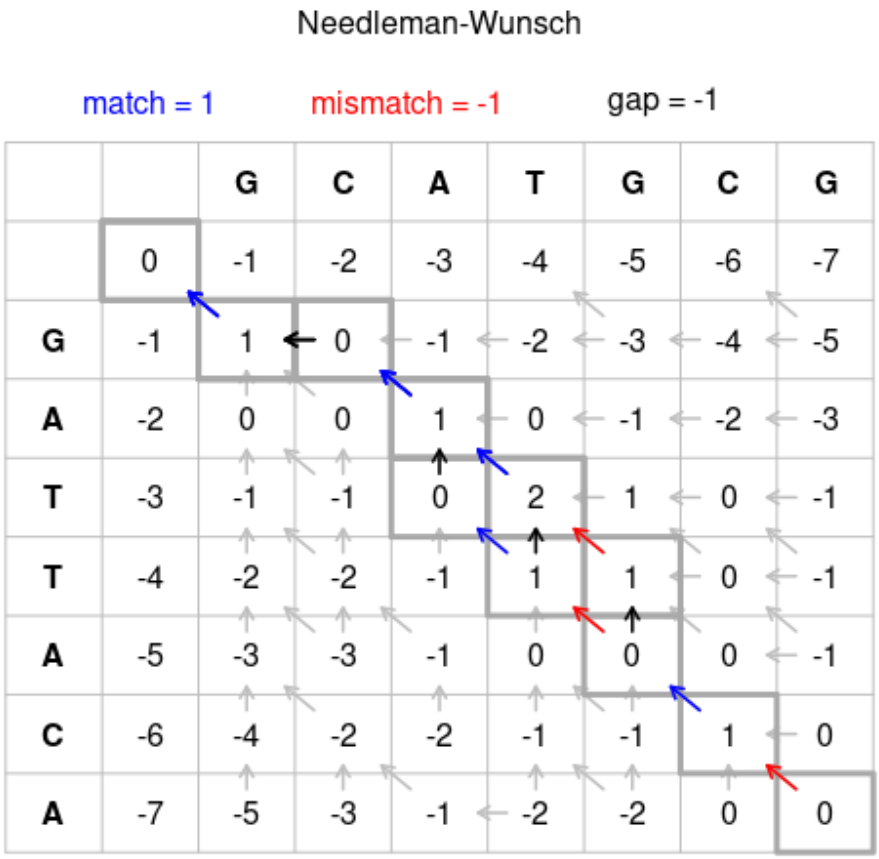


Figure 1. Similarity matrix created with the Needleman-Wunsch algorithm. In this project, the letters will be switched out with coordinate points. Figure obtained from Wikipedia [3]

The first step is to create a similarity matrix of the two trajectories. The similarity matrix is made by calculating the distance between every point of the two trajectories. Figure 1 shows how a similarity matrix of two sequences of letters is made. Start in the top left corner when filling out the matrix, using the cell above, to the left, and diagonally to the top left. The first cell is always 0, as the difference between two empty sequences is nothing. The first row and column take the previous cell and subtract a mismatch penalty. The mismatch penalty is decided based on how ruining the gaps are for the final alignment and will punish the alignments similarity score for having a mismatch. They compare one sequence to nothing,

and the similarity will decrease with the number of letters added. The remaining cells are filled out using the following formula

$$F_{ij} = \max\left(F_{i-1,j-1} + S(A_i, B_j), ; F_{i,j-1} + d, ; F_{i-1,j} + d\right),$$

where $S(A_i, B_j)$ will check if the letters are matching or not and return the value for a match or the value for a mismatch. d represents the gap penalty.

for the cell comparing the first two Gs, the formula will be

$$F_{2,2} = \max\left(F_{1,1} + S(A_2, B_2), ; F_{2,1} - 1, ; F_{1,2} - 1\right)$$

$$F_{2,2} = \max\left(0 + 1, ; -1 - 1, ; -1 - 1\right)$$

$$F_{2,2} = \max\left(1, ; -2, ; -2\right) = 1$$

This step will run recursively until the entire matrix is filled out. Then the second step begins, the backtracking step. That will determine the optimal alignment between the sequences. The value in the cell in the bottom right corner tells us the similarity value. In this example, it is 0. When the values of mismatch and match outweigh each other, there are as many matches as mismatches and gaps.

We start the backtracking in the bottom right corner. The letters are G and A, which is not a match. Therefore, we move to the cell with the highest number, which in this case is diagonal. When it is not a match, moving diagonally in the matrix means we have a gap and can write down both A and G. The following letters are C and C. That is a match, and we move diagonally again and write down both letters. G and A are not a match, so we should move to the cell with the highest number. Here, there are two cells containing 1. That means there are multiple optimal alignments. Which way we choose will conclude with a different optimal alignment. Usually, the alignments will not differ much, but it is easier to get lost in the backtracking step in the case of trajectories. Section 6.4 will discuss further the probability and consequences of multiple alignments.

When adapting this process to the problem of traffic mapping, the entire alignment is not as interesting, only the matches. It means only the coordinates for one of the sequences must be registered whenever there is a match. If matches are identified in one place, the sequences will rarely match again in another place. Thus, this allows for gaps, but the gaps are usually no more than one or two points.

3.1.2. Optimization

Trying to minimize the computation time of Needleman-Wunsch is challenging. The most time-consuming part of the process is the creation of the matrix. Here, every number is dependent on each other, and it is difficult to examine which numbers will be used in the backtracking prior to calculating the numbers. There is some research done in the field of improving performance. They are, however, complex solutions both exploring hardware and software optimization [22]. As it is so complicated, it is not included in this project's scope. For practical reasons, the data went through some optimization instead. These are explained in Section 4.2.1 about preprocessing the data.

3.2. Smith-Waterman

The Smith-Waterman algorithm was inspired by the method of the Needleman-Wunsch algorithm [47]. It is a dynamic programming algorithm identifying the optimal local alignment between two sequences [33]. The algorithm was published in 1981 and was the first algorithm to handle an internal mismatch in an alignment and an internal deletion as well [47]. Like the Needleman-Wunsch algorithm, it provides a scoring of their similarity and how many changes are needed to make one sequence into the other [33]. Unlike the Needleman-Wunsch algorithm, Smith-Waterman only cares about the most similar subsequence, not the entire sequences. In short, it extends the formula described in Section 3.1.1 with a zero to ensure the values in the matrix will never be negative. Alas, Smith-Waterman does not care about subsequences that do not match, only those that match.

That is very similar to the problem in this study. Two trajectories in the dataset will most likely never fully match. That is not what matters. We are only interested in the subsequences that do match. The rest of the sequences are insignificant.

3.2.1. Implementation

Figure 2 shows a similarity matrix created with the Smith-Waterman algorithm. The goal is to find the local alignment and the sub-sequences where the sequences are most similar. There is a value for matches, mismatches, and gaps, like for Needleman-Wunsch. In this example, a match is 3, a mismatch is -2, and a gap is -3. The matrix is calculated the same way as Needleman-Wunsch. However, if the result is negative, the value in the cell will be zero instead. The following formula describes the value for the cells.

$$F_{ij} = \max\left(0, ; F_{i-1,j-1} + S\left(A_i, B_j\right), ; F_{i,j-1} + d, ; F_{i-1,j} + d\right)$$

The only difference from Needleman-Wunsch is the zero that will be considered for the max value. It prevents the value from ever being a negative number.

The backtracking also differs from Needleman-Wunsch. Instead of always beginning in the bottom right corner, Smith-Waterman starts in the cell with the highest value. That allows for only extracting the local alignment with the highest similarity. The remaining backtracking process is executed the same way as Needleman-Wunsch. Matches move diagonally, mismatches move up and to the left, and gaps move diagonally as well. In the end, a result alignment is obtained, which is the locally best match of the two sequences.

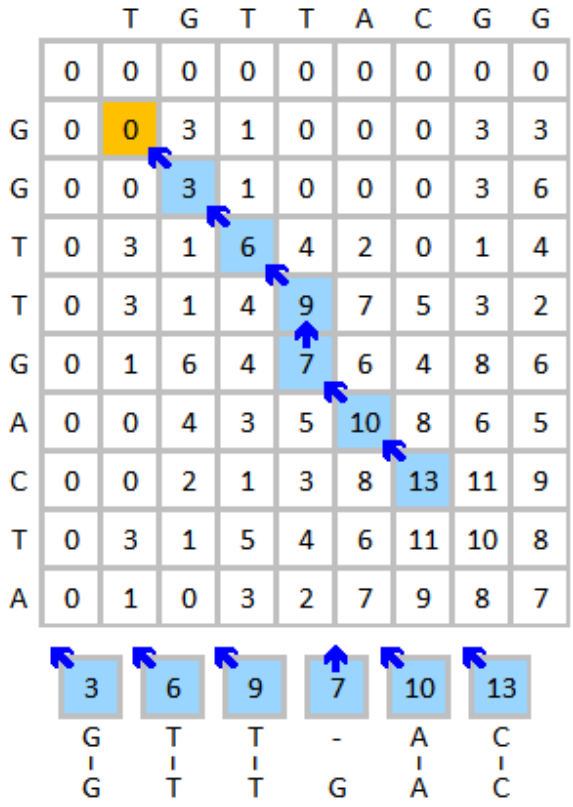


Figure 2. Similarity matrix created with the Smith-Waterman algorithm. Figure obtained from Wikipedia [2]

Considering the definition of the algorithms, Smith-Waterman matches the task best. The problem wants to retrieve the most frequent matching sub-tracks. The clue here is sub-tracks. Unlike Needleman-Wunsch and Dynamic Time Warping, Smith-Waterman is made for retrieving a sub-track. The other two are only modi-

fied to do so. Consequently, Smith-Waterman required the least amount of adaption to fit the problem.

3.2.2. Optimization

This algorithm was modified to reduce the computation time. The running time was just as long as Needleman-Wunsch before the optimization. However, Smith-Waterman opened up to simplifying the creation of the matrix. A matrix between two tracks that are not aligning will only contain zeros. As long as there are no matches, the numbers will not be dependent on each other. When a match appears, and the numbers no longer are only zeros, those will be dependent on each other. That can be exploited to simplify the matrices. Even if two tracks align, the matrix will still consist of many zeros, as they will most likely not match for the entire track. We also know that the maximum distance between two following points in the same track is 20 meters. If the distance between two points is calculated to be 1000 meters, we know that the next point will not be a match either, as the distance will be between 980 and 1020 meters, which is way above the matching threshold of 30 meters. Therefore, if the value for a cell is calculated to be zero, check the distance. The floor of the total distance divided by 40 is the number of cells in one row that can safely be skipped.

This optimization method reduced the computation time drastically. Most tracks that are compared will not match at all. This check made those tracks much faster. Compared to NW and DTW, with a running time of more than 13 hours, Smith-Waterman finished in 1 hour.

3.3. Dynamic Time Warping

Dynamic Time Warping is the only one of the three chosen algorithms frequently used for trajectory similarity. It was initially designed for time series but is today used for everything from speech and image recognition to database indexing and sequence alignment [48, 31]. It is considered a global alignment algorithm like Needleman-Wunsch. Unlike NW and SW, it is based on distance similarity instead of binary matches. Dynamic Time Warping has significantly been explored in the scientific community, however, usually to reduce the computational cost rather than improve the precision [41].

3.3.1. Implementation

Figure 3 shows a similarity matrix created with the Dynamic Time Warping algorithm between two time series. The matrix is created by calculating the distance between the points. That number is added to the smallest number from one of the closest cells. So the distance between the first points of the time series in Figure 3 is zero, as shown in the bottom left corner. The distance between the first point of P and the second point of G is also zero. Getting to the third point of G, the distance to the first point of P is three. The distance to the fourth is nine. This number will be added to the previous three, and the final distance is twelve. This process is repeated for the entire matrix.

The algorithm is similar to Needleman-Wunsch, except that we use the distance between the points instead of the match and mismatch values, making it more applicable to trajectories. Applying it to letters, like Needleman-Wunsch and Smith-Waterman, introduces many challenges and is not recommended.

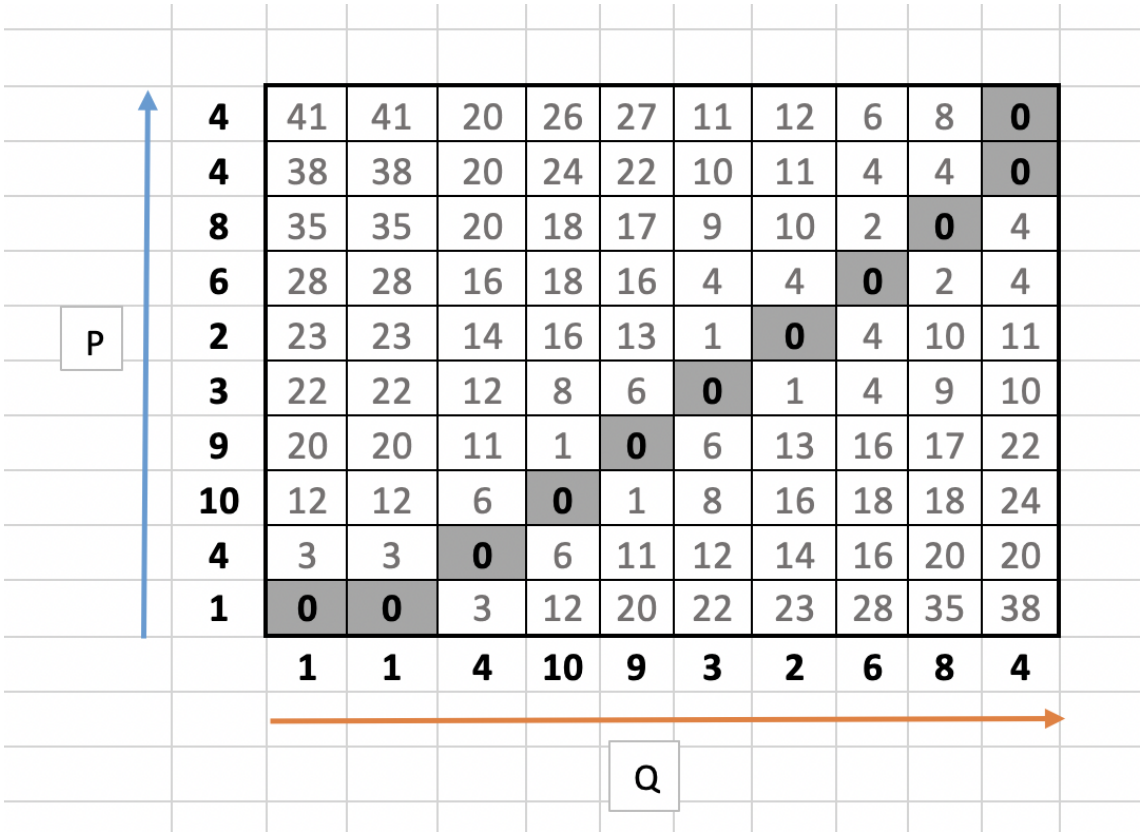


Figure 3. The figure shows a similarity matrix created with Dynamic Time Warping. The grey fields represent the optimal alignment. The figure is obtained from a blogpost on Medium.com [30].

The algorithm was tested using both actual distances in meters and the threshold version that Needleman-Wunsch and Smith-Waterman did. The difference in performance was surprisingly significant. It quickly became apparent that the algorithm was not made for binary numbers. Therefore, the actual distances were used instead.

The backtracking process is straightforward. Start in the corner diagonally across from where we started to fill out the matrix. The Dynamic Time Warping matrix has been written in many different ways, and the corner where we start might not be the bottom right corner. In Figure 3, the start of the backtracking is the top right corner. We follow the lowest numbers from this cell as they lead the path to the most similar alignment. However, that does not apply to getting a local

matching. So for every new cell visited, we also check if the corresponding points are a match. If they are, add them to the alignment. If not, move along.

3.3.2. Optimization

Dynamic Time Warping has difficulty reducing the matrix creation time, similar to Needleman-Wunsch. Numbers are gravely dependent on the previous numbers and will have to be calculated. Therefore, this algorithm is just as dependent on the optimization of the dataset as Needleman-Wunsch. Even though no real optimization adaption was implemented, it does not mean it is impossible. Much research has been done on reducing the computation time for DTW [41, 46, 5]. It is more complex than for SW, but it has been proved possible.

4. Implementation and Experiments

This section includes a description of the dataset used for the experiments. The preprocessing steps performed on the data are also explained. Finally, the experiments to be conducted are presented.

4.1. Dataset

The experiment was conducted using a dataset of taxi data from Porto, Portugal, provided by *UC Irvine School of Information and Computer Science* [1]. This dataset was chosen for several reasons. The taxi data only consisted of trips with passengers traveling from A to B. The original dataset consists of more than 1.7 million taxi trips. However, the dataset owners provided a fraction of the dataset consisting of 320 representative trips. This fraction was used for the experiments due to time limitations. The data also consisted of the entire trajectory, not only start and end positions. They are also created using the same transport vehicle. Some datasets included bicycles and walking trips as well. They cannot, unfortunately, be compared with the same accuracy.

The dataset consists of nine parameters, but only three were relevant for the study:

- Trip_id
- Missing_data
- Polyline

Trip id is used to identify the tracks and is a crucial attribute to the track. Missing data is a true/false value identifying anything wrong with the data and if anything is missing. Polyline is the list of coordinate points that together create the

taxi trip. Figure 4 displays a heatmap of the 320 tracks used for this study. It can provide some insight into the results, but it is not accurate enough to tell exactly.

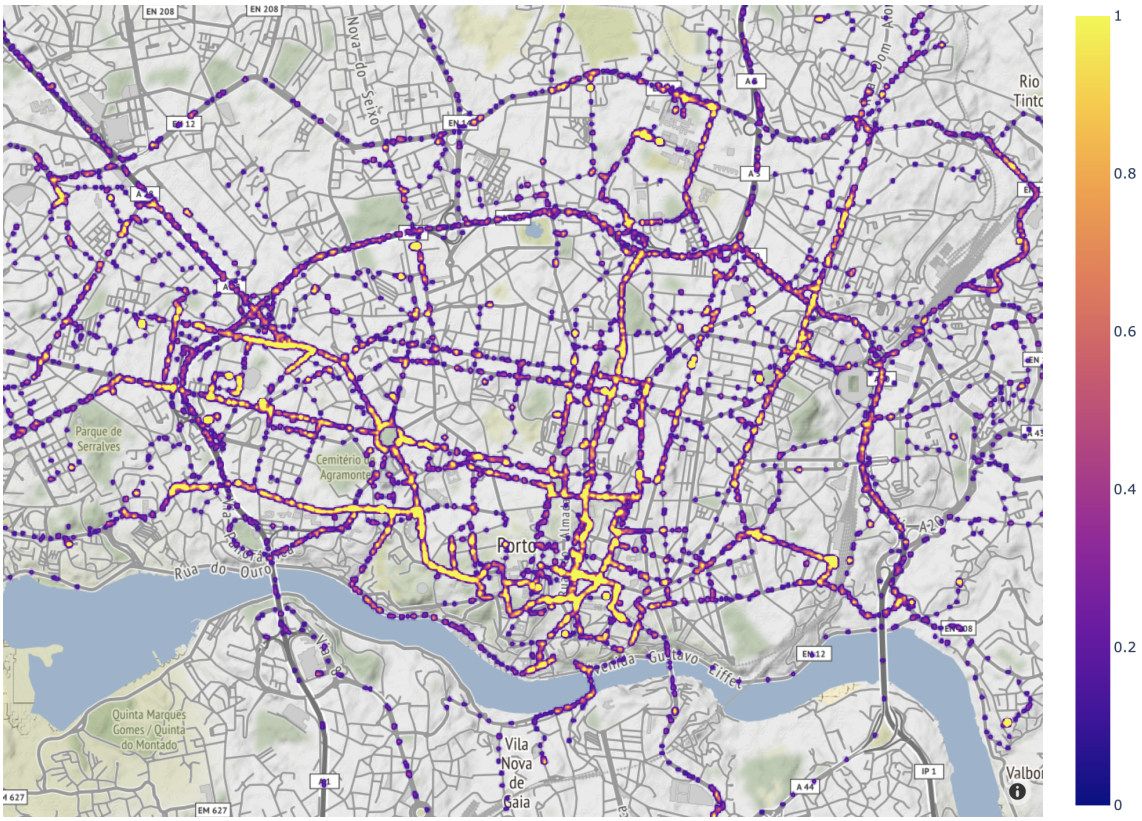


Figure 4. Heatmap constructed from 320 records of the taxi dataset from Porto, Portugal

4.2. Preprocessing

The data had to be preprocessed before being used for the experiments. The dataset had a column called "Missing data" with a true or false value. The tracks could also have an empty polyline field. These steps removed no tracks in the partial dataset used for the experiments. Polylines with three or fewer coordinate points were also deleted. They do not represent a valid taxi trip and cannot provide information about people's normal transportation behavior. Eight tracks met this criterion and were removed.

The data is collected by transmitting a coordinate position every few seconds. That means the distance between two points can vary a lot depending on the speed of the taxi and the GPS signals. When calculating if two points are a match or not, they might be driving on the same road, but their GPS is transmitting positional data at different places. Even though their vector is parallel, these places could be 50-60 meters apart and would not qualify as a match. Figure 5 shows two tracks in the left image. The right image shows the alignment between the tracks created with Smith-Waterman. We can see the alignment of the beginning of the tracks is found. However, when the tracks move onto the larger road, the points are further apart, and the algorithm cannot find an accurate match. The image also shows how a track can skip an intersection and jump straight to another road. The direction of the polyline will not be the same, and the tracks will no longer be parallel. Another aspect that could be a problem is a large highway with three or four lanes or more. It is not a problem in a small city such as Porto, but in a larger city with wider highways, there could be more than 20 meters between two cars driving the same road.

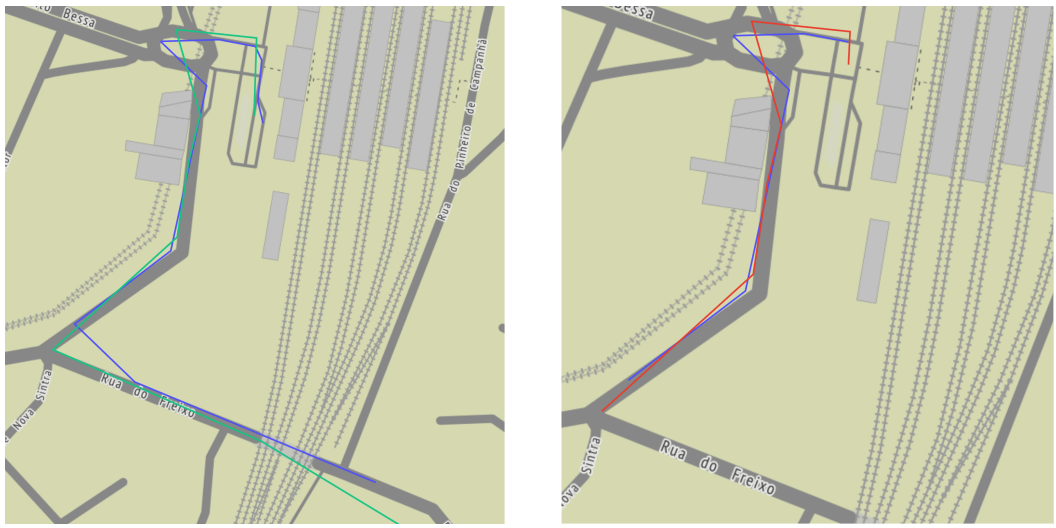


Figure 5. The image to the left shows two tracks. They overlap until the blue track stops, and the green continues. This is prior to adding additional coordinate points. The image to the right shows the alignment of the two tracks created with Smith-Waterman. The algorithm identified the alignment at the beginning of the tracks. However, it ends before the intersection, and the alignment on Rua do Freixo is not represented. That could be due to another speed limit on this road, and points occur less frequently. Therefore the algorithm had difficulties identifying these tracks as matching on this road.

That is a common problem when dealing with GPS trajectories. It can be solved by using vectors for comparison instead of single points to identify matches. When points are too far away, their vectors will match instead. However, this complicates the process severely. Comparing vectors is heavier than computing a distance. That is why the most common solution to this problem, and what was decided to do here, was to add waypoints in between the existing points.

The waypoints were added based on two needs. Firstly, the threshold to define two points as a match or not was set to be 30 meters. Therefore, the distance between two points in the polyline should be less than 30 meters. Secondly, too many waypoints should be avoided as the size of the matrices will expand, and the computation time will also be extended. Twenty meters turned out to be a safe number to

meet both needs. Figure 6 shows the alignment between the tracks from Figure 5 after adding waypoints. We can see here that the entire overlap is found.

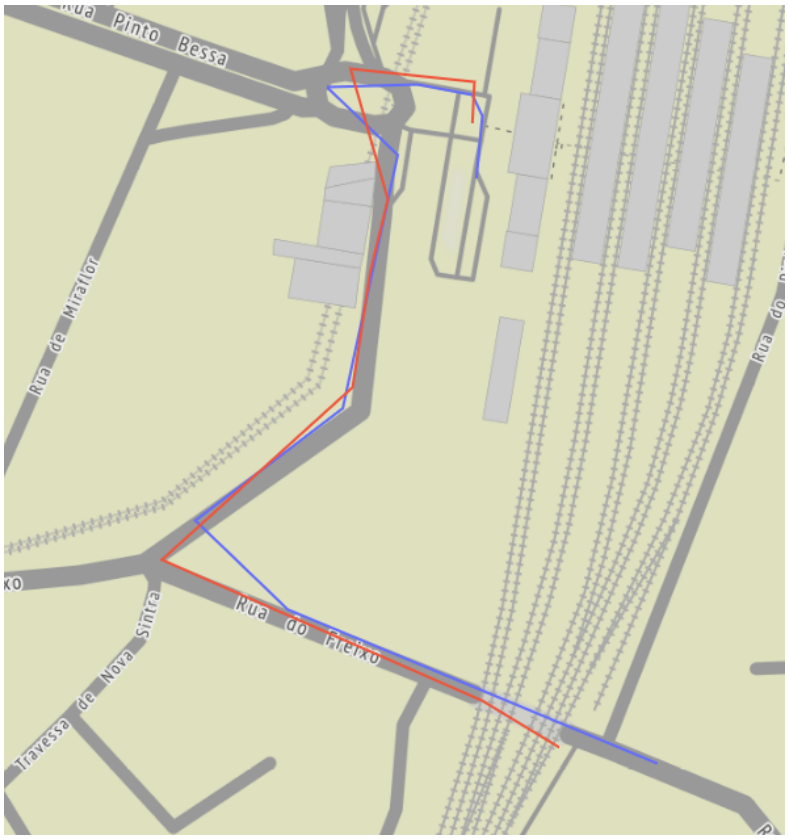


Figure 6. The image shows the alignment between the two tracks after waypoints were added between existing points.

4.2.1. Optimization

This data was enough to make the Smith-Waterman algorithm run effectively. For the other two, the runtime was still unpractically long. Two preprocessing steps were added to make them run faster. However, when comparing the result of three different algorithms, the input should be the same to compare their output correctly. That is why Smith-Waterman will also be using the same preprocessed data as Needleman-Wunsch and Dynamic Time Warping.

The first step to shorten the running time was to remove the points outside the city center. The heatmap of the dataset shows that most tracks are located inside

the city center. Therefore, there is no need to include these points in the calculation. It was considered not only removing the points outside the city center but entire tracks with points outside the city center. However, we want a mapping of the natural traffic in the city, and removing tracks going outside the center would remove many natural and typical tracks. Nevertheless, all points outside of the city center were removed. The city center is defined individually for each city. For Porto, it was decided based on the heatmap of the tracks. It is visible where the primary traffic is situated. Figure 7 shows the heatmap with a square. The points inside the square were kept, and the points outside it were discarded.

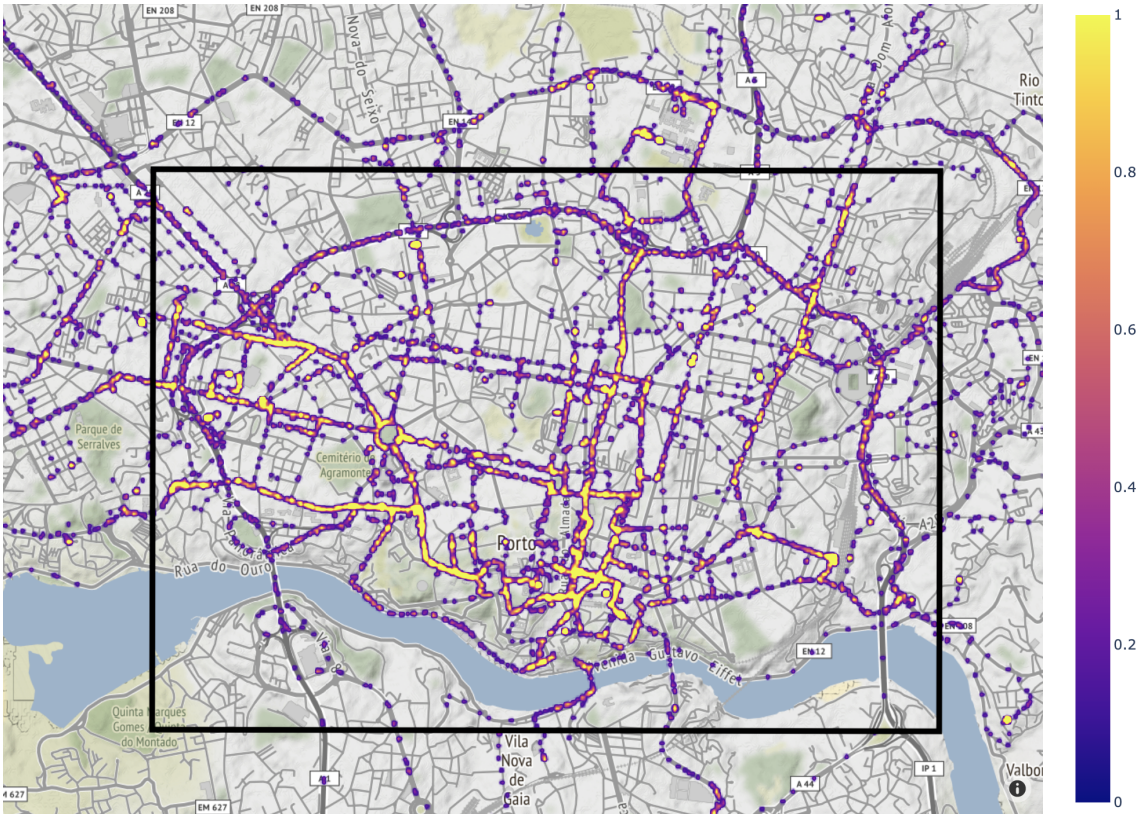


Figure 7. The figure shows the heatmap from Figure 4 with a square. The coordinate points outside the square were discarded.

The second step to reduce the computation time was to divide the tracks into subtracks. That is to reduce the length of the longest tracks to minimize the size of the similarity matrices. Even if more matrices are to be made, they would be smaller and faster to create. Where to divide the tracks was a problem to be

solved. If the tracks were divided in the middle or a random place, we would risk dividing a long frequent result track. That could have a considerable negative impact on the result that would not be justified by the shorter running time. It would be better to divide the tracks between these longer, frequent sub-tracks. We want to divide where points are less frequent. To identify the fields with less frequent points, we use DBSCAN. DBSCAN is a clustering algorithm for clustering points based on position. What DBSCAN also does is identify noise points. Points that do not belong in a cluster are identified as noise. These noise points will identify the fields with less frequent points. That is where the division should be made. The noise points are removed, and the sub-tracks before, after, or between will be considered separate tracks. The dataset will now consist of more tracks, fewer points, and smaller matrices. Together, these two preprocessing steps reduced the computation time by 75 %.

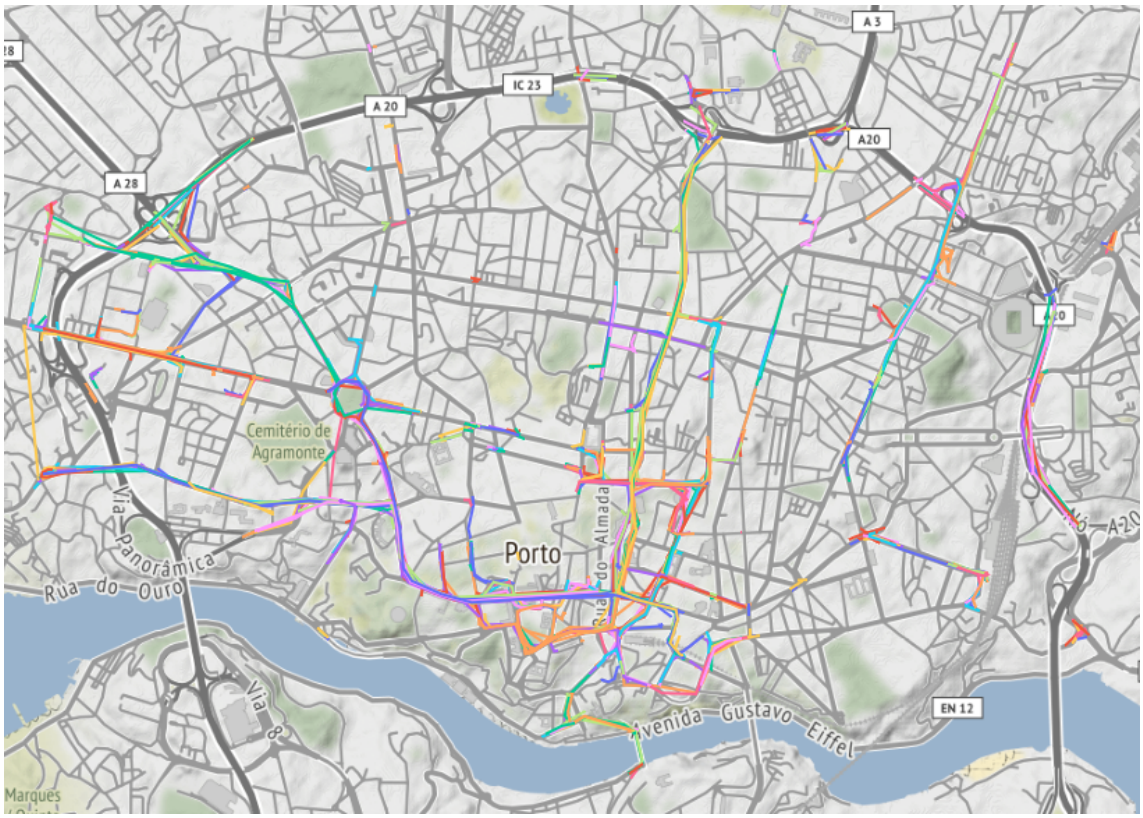


Figure 8. When noise is removed, sub-tracks can be created from the resulting coordinate points.

4.3. Creating a Result List

The study's goal is to obtain an ordered list of the top most trafficked roads in a city. The result list consists of the polylines of the most represented sub-tracks and a score determining their ordering in the list. Obtaining a result list from the output of the algorithms is not directly a part of the study and will not be evaluated. Nevertheless, the method will be explained in this section.

Every track in the dataset has to be compared to every other track. This means that for every track, a matrix and backtrack alignment will be created with every other track. Track 1 will be compared to every track except itself. Track 2 will be compared to every track except track 1 and itself. When we come to track 20, it only has to be compared to tracks 21, 22, and so on, as it has already been compared to tracks 1, 2, and up to 19. Every track will generate a list of alignments that match some part of the track. The alignment is not included if it is shorter than four coordinate points. For every coordinate point in the track, there is a count for how many other tracks overlap with that point. This is the frequency. We must first evaluate the weighting between length and frequency to decide which alignments belong in the result list. The length in this formula is the number of coordinate points. Every coordinate point is approximately 20 meters apart. We do not want the longest track if it means it occurs three times, as it is long but not frequent. At the same time, we do not want tracks with two coordinate points but with a high frequency, as there is a higher likelihood of inaccuracy. The formula was found by trying and failing to generate the best possible result. It is adapted to this dataset and would not necessarily work for other datasets. The formula was decided to be the length in coordinate points plus frequency. The reason behind this decision is further explored in Section 6.3

The potential resulting alignments will receive a score based on this calculation. These alignments are different combinations of coordinate points to maximize the score. Will adding three more points to the alignment generate a higher score if

the frequency will have to decrease with two? The alignment or alignments with the highest score will be added to the final result list. A track can generate multiple resulting top alignments if they do not overlap. When all tracks have been compared to all other tracks, and some have generated one or more result tracks, the result list is sorted based on the tracks' scores. Tracks with less than four coordinate points or frequency below eight were excluded from the result list. The final result list is, thus, a list of polylines sorted on a score determining how relevant the polyline is to the problem.

4.4. Experiments

The experiments in this study will analyze the results from the algorithms. The reliability and quality of the results will also be tested and explored. That is to provide insights into differences and similarities that will be discussed and evaluated in Section 6 Discussion.

The first experiment is to compare the result lists. They will also be compared to the heatmap created from the same preprocessed dataset. The heatmap cannot work as a solution as it only includes the coordinate points and not the lines between them. Points cannot be blindly trusted as they will, for example, not register any in tunnels. Larger highways with a high speed limit will generate fewer points, as the cars are driving faster and the distance between points will be longer. Even though more cars are driving these roads, they might appear as bright in the heatmap as one taxi driving in circles on a small side road trying to park, turn around or wait for a passenger. The heatmap does not know which points belong to the same track either. The reasons above could result in bright spots on the heatmap even though all the points originate from the same track.

Comparing the result lists to a correct solution is therefore tricky. The top 10-lists, the top 20-lists, and the entire lists will be compared to each other and the heatmap to identify differences and similarities.

The scoring of the result tracks is a combination of length and frequency. Both a high length and a high frequency are positive traits of the result. It means the algorithms have been able to find long tracks with many matches. Therefore, the top scores and average length and frequency will be compared to identify the differences in the algorithms.

The length of the result lists will also provide insight into how many matches and valuable result tracks the algorithms were able to find. The threshold for a track to be included in the result list is the same for all algorithms. A longer list will, therefore, mean more matches have been found.

The last experiment will delve deeper into the details of discovering a match. All three algorithms will be tested on only one of the tracks. They will compare track 2 to every other track in the dataset, and the matches they find will be compared. How many matching tracks were discovered? Which tracks were not discovered and why? For example, how long are the matches not discovered by any or all of the algorithms? It will not include creating the result list, so only the algorithms will be evaluated. That will indicate something about how sensitive the algorithms are to matches. Will they need a long match to identify them, or will they discover even the more minor matches? Alternatively, some algorithms might miss a more significant match or maybe not discover the entire match, only a fraction of it. These results could also help explain the results of the other experiments due to investigating deeper how the algorithms work.

5. Results

This chapter will present the results of the experiments explained in Section 4.4. The next chapter will discuss them and provide more insight into the essence of the results.

5.1. Result Lists

The first experiment is comparing result lists. The complete lists are included in the appendix, along with the code Section A.1. Figure 10, Figure 11 and Figure 12 display the maps of the result tracks from the entire result lists. As can be seen, many of the result tracks overlap and cover the same roads. That is because the roads included in a result list of track 1 are still included in track 3. It means track 1 will output a result sequence that matches many other tracks. When track 3 is compared to the rest of the tracks, it might output the exact same result track, but with a frequency lower by one. Figure 9 displays three tracks that overlap entirely. The first of them will match with all three. The second will not be compared to the first, thus only getting one match. The third will not match anyone, but it will still be represented through the other two. That is why the result list is full of overlapping tracks. It can also be a measure of how frequent that road segment is.

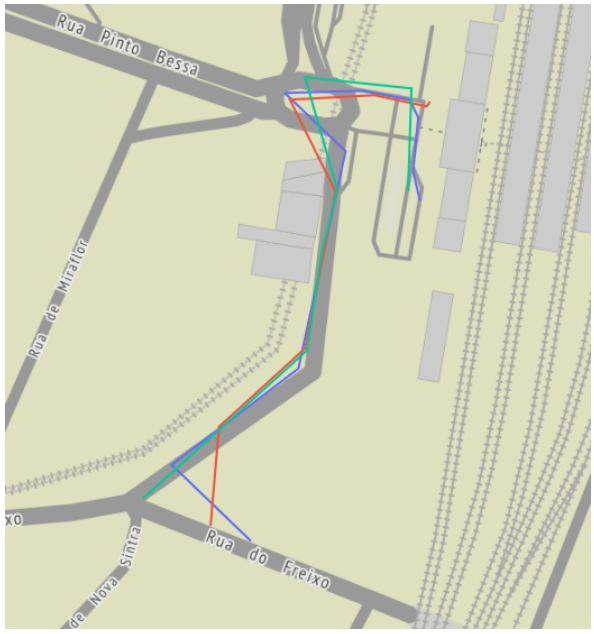


Figure 9. Plot of three overlapping tracks

The complete result lists of all three algorithms are relatively similar. They cover the same main roads. When looking closer, some minimal differences can be seen. However, nothing can be read from them as they are so minimal. They are short tracks in vaguely the same area and can be considered the same.

Figure 10. Complete result from the Needleman-Wunsch algorithm

Figure 11. Complete result from the Smith-Waterman algorithm

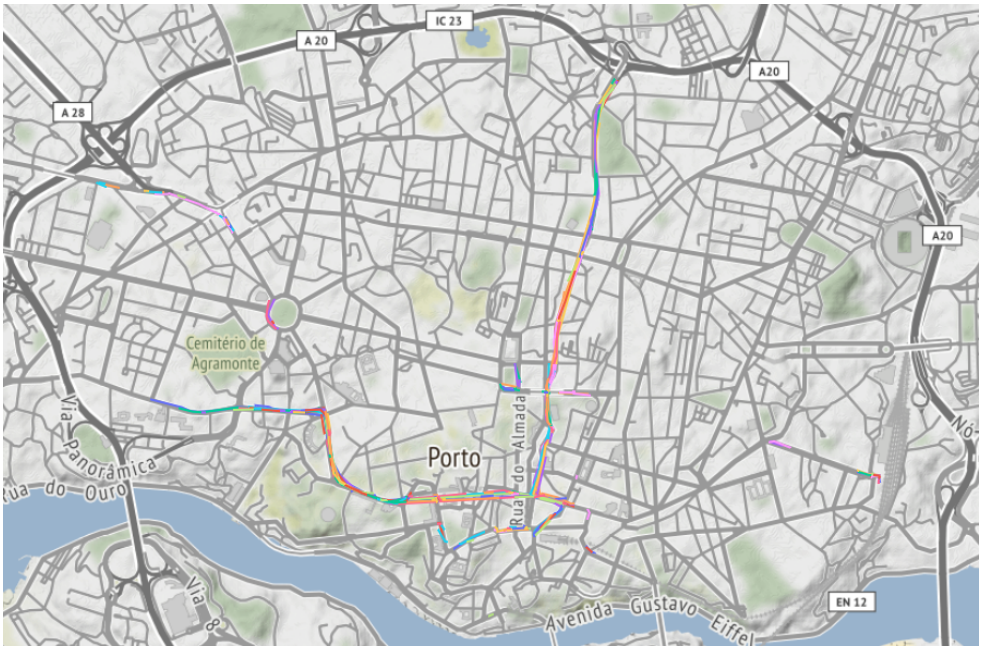


Figure 12. Complete result from the Dynamic Time Warping algorithm

The entire result list is not a very accurate solution to the problem in this thesis. The study looks for the two, three, or maybe four topmost trafficked roads. The entire list provides more than that, as some tracks might end up in the list by coincidence. Many of these outliers are filtered out in the top 20 result lists. What remains could be the longest, most trafficked roads. Figure 13, Figure 14 and Figure 15 displays the top-20 result lists of the three algorithms. They cover very much of the same ground and only the most significant roads from the entire list. The only noticeable difference that can be seen is on the road marked by an A. The resulting road for NW and SW stops before DTW. This road for DTW continues for as long as the entire result list.

Figure 13. Top-20 result list from the Needleman-Wunsch algorithm

Figure 14. Top-20 result list from the Smith-Waterman algorithm

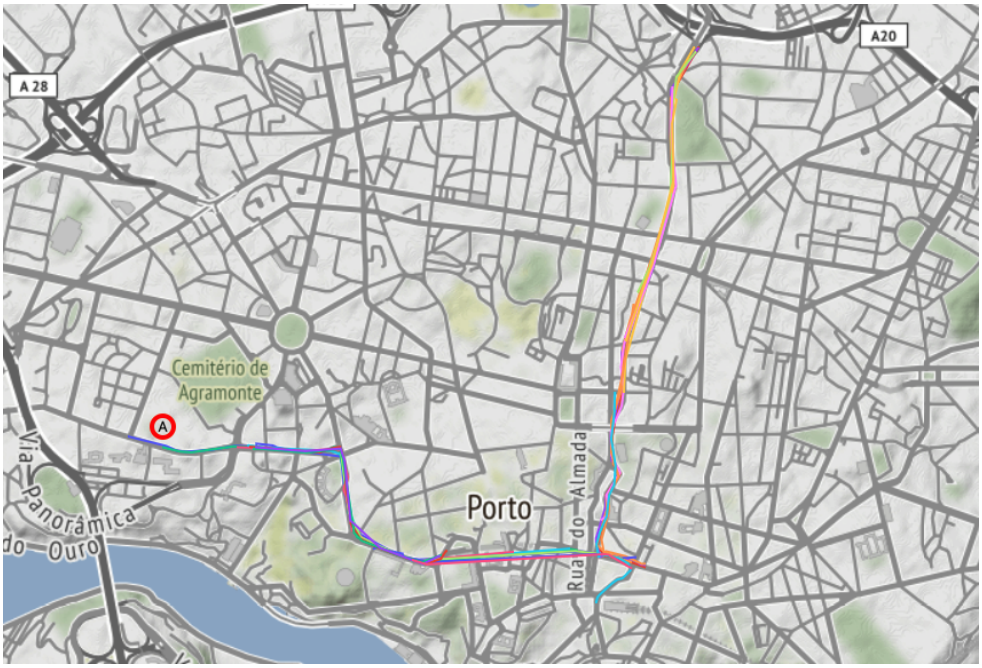


Figure 15. Top-20 result list from the Dynamic Time Warping algorithm

If we filter out even more and only look at the top 10 result list, we get the maps shown in Figure 16, Figure 17 and Figure 18. They are very similar to the top 20 lists, but something has changed. DTW still has the longest road marked by an A. Although, SW and NW have kept the road marked by a B. DTW lost this road in the top-10 list.

Figure 16. Top-10 result list from the Needleman-Wunsch algorithm

Figure 17. Top-10 result list from the Smith-Waterman algorithm

Figure 18. Top-10 result list from the Dynamic Time Warping algorithm

5.2. Average Length of Result Tracks

The average length of result tracks is shown in Table 1. The length is measured in the number of coordinate points included in the track. DTW has the longest tracks in all three lists. NW and SW are a bit shorter but very similar between them. The higher on the result list, the more significant the difference is between DTW and the others. However, the lengths are overall very similar.

Table 1. The table shows the average length of the result tracks in the corresponding result list.

	Smith-Waterman	**Needleman-Wunsch**	**Dynamic Time Warping**
Top-10	130,7	130,6	138,6
Top-20	103,7	103,55	108,75
Entire list	24,43	25,15	24,69

5.3. Score

The scores of the result tracks are calculated by adding length and frequency. The result list is sorted by this score, where a higher score means a longer and/or more frequent result track. Smith-Waterman discovered a result track with a score of 170. Needleman-Wunsch found a track scoring 169. Dynamic Time Warping got a score of 191. Thus, the highest score was found by Dynamic Time Warping, which either found a longer track than the other two or found more tracks matching the resulting track.

5.4. Length of Result List

The length of the result list says something about how many matching sub-tracks the algorithm has discovered. Although, it must be seen in the context of the other results. Smith-Waterman found 146 result tracks, Needleman-Wunsch found 138, and Dynamic Time Warping found 150 result tracks.

5.5. Average Frequency

The average frequency for the different lists will tell us how many matching tracks the algorithms find for each result track. Table 2 displays the average frequencies of the result lists. These numbers mean the average number of tracks going through a point represented in a track from the respective result list.

Table 2. The table displays the average frequencies found by the three algorithms.

	Smith-Waterman	**Needleman-Wunsch**	**Dynamic Time Warping**
Top-10	10	9,8	13,4
Top-20	9,85	9,75	12,2
Entire list	10,62	10,49	10,33

5.6. Total Amount of Coordinate Points

Another number that will help us understand the results is the total number of coordinate points in the result lists. Smith-Waterman found, in total, 3567 coordinate points, Needleman-Wunsch found 3471 points, and Dynamic Time Warping found 3704. Naturally, DTW found the most coordinate points as their result list is longer than the others. Other reasons why will be brought up in Section 6.

5.7. Qualitative Comparison of Algorithm Features

This experiment details how well the algorithm works for a single track. It only uses the implementation of the algorithms and will not be creating a result list or calculating a score. The experiment will provide a more detailed impression of the algorithms. The focus is on how many matching sub-tracks the algorithm can find. The algorithms' traits will be explored to better prepare for discussing which one fits the problem the best.

The experiment was primarily conducted on track 2 from the dataset. It means all three algorithms compared track 2 to all the other tracks. Other tracks were also checked to verify the results from track 2. Table 3 shows the matching tracks only one or two of the algorithms was able to find for track 2. A total of 23 tracks were found by all three algorithms. They are not included in this table. The complete table of all matching tracks found is included in Section A.2. SW identified 29 matching tracks while both DTW and NW found 25. SW found 529 matching points, DTW found 513 points, and NW found 510 points.

The track-ids with a longer track-id than the standard numbers 1-327 are generated when splitting the tracks into sub-tracks. Track 51671577 is a sub-track of track 51, which had to be given a unique id.

Table 3. Table showing which algorithm identified which matching tracks

Smith-Waterman	**Needleman-Wunsch**	**Dynamic Time Warping**
T18	T18	
	T51671577	
T161674001		
		T18014434
T190		
T194		
T207675434		
T221		
		T258

Every track in the table except two consists of 4 coordinate points. The limit is 3, which means every algorithm might have found them, but not enough points to score above the threshold. The two tracks with more points are T18 and T190, with 13 and 10 coordinate points, respectively.

To confirm these numbers and make sure this is not an exception but rather a representative track from the dataset, a couple of other tracks were also checked, if not as thorough. Track 44 was tested. SW found 49 matching tracks and a total of 2780 points NW found 48 matches and 2831 points. DTW found 49 matches and 2789 points. All matching tracks that differ between the result lists are around three to four coordinate points. These are probably tracks going the opposite direction or crossing track 44. They are not significant.

Track 32 was checked as well. Here, SW found 47 matches with a total of 1334 coordinate points. NW found 43 matches and 1300 points. DTW found 45 matches with 1365 points. Smith-Waterman found a considerably higher amount of matches for this track, although DTW still found more points. Two of the tracks NW did not find, and one DTW did not find were long tracks they should have identified as a match.

6. Discussion

This section will discuss and evaluate the results presented in Section 5. Other metrics that cannot be measured in numbers or values will also be drawn into the discussion. This chapter will explore the reasons behind the results. It is essential to understand that the experiments should be seen in context to each other and understand what might affect them to be what they are.

6.1. Comparing the Result Lists

The first impression of the result lists is that they are very similar. All three algorithms have found approximately the same roads. The two roads represented in the top 10 lists are visible in all the result maps. As all the algorithms output the same roads, there is a higher probability that these results are, in fact, the most frequent roads. Many things can affect the final result list, so the intention was to look at the top 10-list at most. However, when checked, it turned out that the two top roads were the same for all three algorithms. All top 20 lists only consist of these two roads or fractions of these two roads. As the results are so unanimous, it would not be too bold to suggest these are the two most trafficked roads in Porto, based on the dataset and the method used.

The heatmap in Figure 19 is made from the same dataset used in the experiments. It is only based on the density of points and not the tracks they belong to. The heatmap also shows the two roads represented in the results. Generally speaking, when comparing this map to the complete result lists from all three algorithms, we can clearly see that the same roads stand out. Almost all prominent bright yellow parts are represented in the results. Only one road on the heatmap should have been in the results lists. The road is marked by a C on the heatmap. This road is not fully represented in any result list, but it is still a long bright line on the heatmap. Looking very closely at the resulting maps from Smith-Waterman and

Needleman-Wunsch, a tiny dot can be seen on the road. It is difficult to say why it is not better represented. It could be a few tracks that have generated many coordinate points. Alternatively, too few tracks move in the same direction. As will be better explained later in this chapter, tracks going in opposite directions will not match or only generate a very short match. These are not visible metrics on a heatmap but will count in the results.

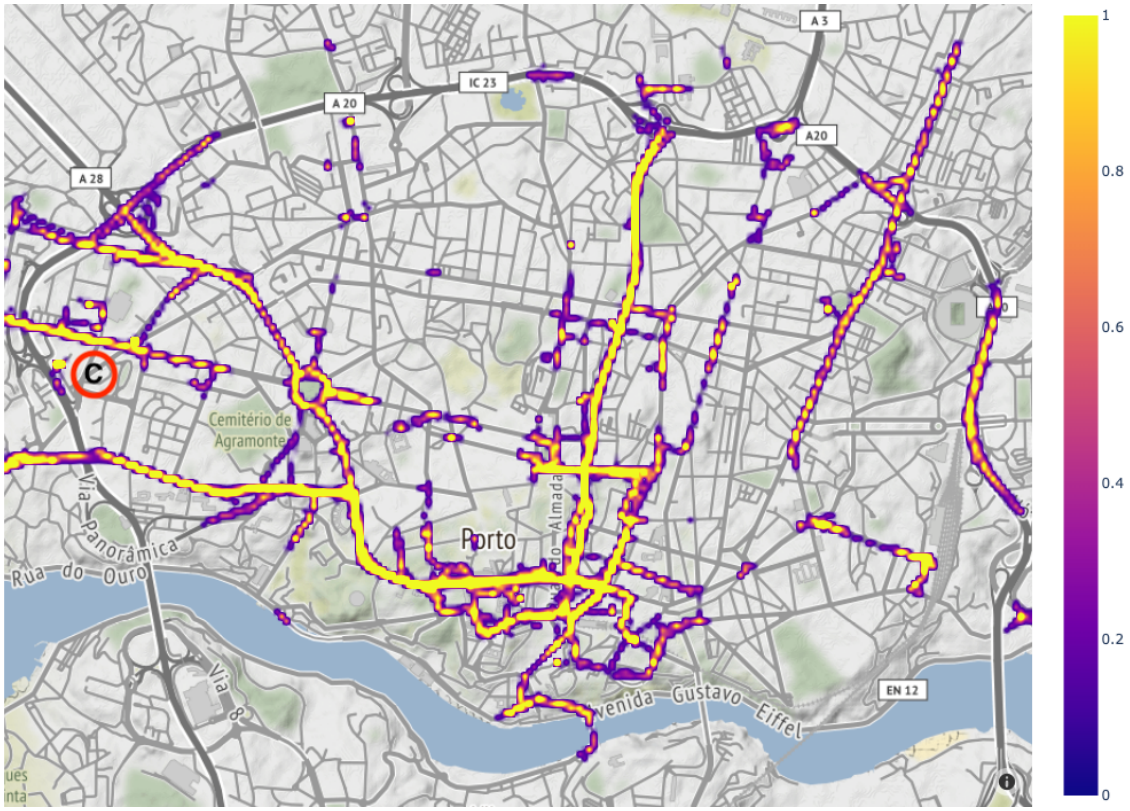


Figure 19. Heatmap created from the preprocessed dataset the algorithms were run on.

The further down on the result list, the more they differ, and the less interesting the tracks get. They could be represented by coincidence and is not necessarily a part of the most represented sub-trajectories in the dataset. Therefore, the top part of the list is the most significant. However, it is interesting that all three result maps look very much alike. Only the shortest tracks differ slightly. The fact that the maps are very similar is a quality assurance for the results. When they all found such similar roads, the chance of it being a good result is higher. Although,

we must remember that the three algorithms are also very similar. Therefore, they might have made the same mistakes leading to similar results. It could also be inaccurate because the dataset is not representative, or the way alignments are gathered into result tracks is not good enough. Nonetheless, if we consider the comparison to the heatmap and presume the dataset is representative, we can establish that the algorithms have identified the most trafficked roads in Porto. They might not be the most suited algorithms for this problem, but they provided a satisfactory result.

6.2. Length and Scoring of Result Tracks

The length of the result list says something about how many matching tracks and points the algorithms found. All matches count as valid matches of tracks that do overlap. There are occurrences where matches are registered when they should not. It can, for example, be two tracks crossing each other, where barely enough points will be matching for it to count as a correct alignment. Something that we will return to in Section 6.4 is the case where two tracks might match one place and cross each other later. They will match twice even if the crossing only matches one or two points. That only happens with Needleman-Wunsch and Dynamic Time Warping. As explained, the reasons behind a result can be numerous, and therefore it is difficult to conclude anything from them isolated. They have to be seen in the context of the other numbers. If the result list is long, it might have found many matching tracks. Alternatively, it might have missed some points and divided all result tracks into two.

Dynamic Time Warping scores the highest numbers in almost every result from the length and scoring category. It has the longest list with the most coordinate points and the highest score. It also has the highest average length and frequency of the top parts of the result list. However, this does not automatically mean DTW is the best suited. Nevertheless, we do know DTW has found the most coordinate point matches.

6.3. The Scoring Formula

When calculating the average length of result tracks, it became clear that the scoring formula greatly favors longer tracks. The score is calculated by taking the length and adding the frequency. The frequency is defined by how many times the sub-track is represented in the dataset. The length is the number of coordinate points creating the track. The scoring formula can easily be changed and adapted to whatever suits the problem the best. Tracks are added to the result list based on the score. It might favor adding longer, rarer tracks than short, frequent ones if the length has too much impact. It is a problem that a long road where a few cars drive undoubtedly will be at the top of the list. It might not be the most frequent, but it is a long one.

Looking closer at the complete result list, it is clear that the tracks with the highest frequencies are often the shorter tracks. The very frequent tracks keep to the middle part of the list as they are not long enough to score higher. It was not the intention for the result list to almost only be ordered by length, but doubling or tripling the frequency did not affect the result list too much. It only shortened some result tracks with a couple of coordinates favoring a higher frequency. Although, the average frequency still stayed around 10. A couple of tracks also switched places, but the top 10 list consisted of the same tracks. When only including frequency in calculating the score, some changes happened. The tracks became much shorter, with an average of 5 coordinate points. Nevertheless, the result tracks were still located in the same area. A map of this result is included in Section A.3 in the Appendix. The only significant difference is an outlier to the far east that maybe should have been included in the top roads. It is marked with a D on the maps in Section A.3. However, it is very short, so it depends on the application of the result.

6.4. Qualitative Comparison of Algorithm Features

The experiments exploring the three alignment algorithms' results on a single track discovered similar results. Smith-Waterman found a few more tracks matching track 2 than the other two algorithms. To investigate how good or bad the algorithms were, the tracks that should have been found were identified. All the algorithms found most of these tracks. A total of seven tracks that appeared to be overlapping were not represented in the result lists. It turned out the reason was due to the direction of the track. A feature of the algorithms is that they only identify the tracks moving in the same direction. Cars driving in the opposite direction on the same road will probably only be able to output 2 or 3 matching points which are less than the requirement to qualify as a matching track.

Out of the nine tracks not found by all algorithms, five were in the wrong direction and only matched with a few points for some of the algorithms. Two were found by SW, two by DTW, and one by NW. Two more tracks were only crossing the original track and thus, should not match either, but Smith-Waterman achieved to match it with four of its points. Then the last two are the interesting ones. T18 and T190 had a 13- and 10-point match, respectively. Smith-Waterman identified both, and Needleman-Wunsch found one of them. For T18, DTW got confused by the long tracks. If shortening the track, DTW could identify a match, but it got lost in the backtracking for the complete track and missed the matching. T190 was only identified by Smith-Waterman. For Needleman-Wunsch, the same thing happened. The tracks were too long, and the backtracking got lost. For DTW, another incident occurred. Duplicate points are removed to avoid false matches. For track 2, four redundant coordinate points were deleted. It resulted in DTW not being able to identify the match with T190. DTW did find it when the duplicate points were present and when the track was shortened. It indicates that the backtracking for DTW is unreliable and cannot be trusted.

The reason why NW and DTW get lost in the backtracking sometimes is due to multiple optimal alignments. As explained in Section 3.1.1, using coordinate points instead of letters makes the matrix act differently. They follow the backtracking in a greedy approach. The result is that they cannot guarantee that enough matching points will be a part of the identified alignment.

A trait became apparent with the matching between tracks 32 and 130. Figure 20 shows the two tracks, T32 and T130, to the left and the three matching alignments to the right. Smith-Waterman, the green line, only outputs the alignment far south where the tracks actually overlap. DTW and NW also find these matches. However, when the tracks overlap again further north, they find a couple of more matches that would not matter if other matches were not already found. Because the total amount of matching coordinate points will exceed the threshold of 3, all points will be included. Smith-Waterman is a local alignment algorithm that stops backtracking when the traceback from the highest number reaches zero. Needleman-Wunsch and Dynamic Time Warping do not stop. They include every matching point, even if there are many points in between where they do not match. They open up for the case where two tracks can match more than once. Smith-Waterman only outputs the longest matching alignment, even if they overlap more than once. This is not a frequent incident, but it can occur.

Figure 20. The left image shows track 32 in red and track 130 in pink. The right image shows the three alignments found between the two tracks. Smith-Waterman in green, Needleman-Wunsch in red, and Dynamic Time Warping in blue.

Another trait became apparent for track 32. It might originate from what happened between tracks 32 and 130. For the entire track 32, Smith-Waterman identified two result tracks, one track with a score of 57 and one with a score of 41. Their lengths were 41 and 30, respectively. Dynamic Time Warping found only one track, but it received a score of 86 with a length of 77. This incident might have happened because DTW has identified more matching points due to the reason explained in the last paragraph. DTW could have filled in points in between two matching lengths, or maybe SW failed to locate more matches. Nonetheless, the frequencies of Smith-Waterman's matches were 16 and 11. DTW only had a frequency of 9. This incident could have consequences for the solution, as the difference in score is so large that they will be far from each other in the result list. Which of them is correct is difficult to determine. However, due to the results discussed in this chapter, it might be possible that DTW has included more points than it should.

6.5. Implementation and Metrics

The implementation and the method of the algorithms are very similar. They all create a similarity matrix between every pair of tracks, backtrack the matrix, and output the matching sub-track if they match. They are still different in creating the matrix and the backtracking strategy.

As explained, Smith-Waterman can handle a larger and more complex dataset than the other two. That is because the matrix between two tracks that do not overlap only consists of zeros, and the values are, thus, not dependent on previous numbers. When a match occurs, the numbers are dependent on each other again, as the values will no longer be zero. That makes the implementation of Smith-Waterman a lot easier, which again helps to shorten the computation time.

Dynamic Time Warping is the only one of the algorithms made for time series or trajectories following a path in time. It makes the algorithm more suitable for the data, and less adaption needs to be made. However, it is an algorithm explicitly made to recognize the similarity in movement and waves along an axis. It is not made for vessels on a map moving back and forth. When backtracking, DTW makes the greedy choice of moving towards the smallest number. For the trajectories moving along a road network, the probability for the greedy choice to be the best choice is lower.

Needleman-Wunsch has not been significantly researched for handling movement data or reducing computation time. However, it has been used more for GPS data than Smith-Waterman. When identifying the most trafficked road based on taxi trips, we are not interested in the full tracks but the most frequent sub-tracks. That is, by definition, a local alignment problem, and thus, Smith-Waterman should perform better than Needleman-Wunsch. The NW algorithm was implemented in a way to output the local alignment, but because it is not what the algorithm is made for, it is a fundamental disadvantage.

6.6. Application

Now that we have established the algorithms work for trajectory similarity, we should discuss their appliance. As mentioned in Section 1.1, this method has many applications. There are many practical usabilities of the result lists of the most trafficked roads. Other applications can be connected to the usage of the alignment algorithms for trajectory similarity and comparison. The map or list of the most trafficked roads can be used to solve transportation problems, routing traffic, and renovate roads. The results of the algorithms show that they can be applied to problems regarding comparison and similarity between trajectories.

The map of the most represented sub-tracks shows which roads suffer the most load. These roads should be known when building schools, playgrounds, or other places there are likely to be kids. There should be safe ways to cross the roads and safe places to follow the roads on foot or by bike. When creating public transportation routes or building cycling fields, these roads should be in focus.

The traffic map would also be helpful for mapping air and noise pollution. Much research has been done in this field that can help city planners or house builders in their work. City planners should build noise barriers to protect the citizens from noise. Environmental measures should be prioritized in highly trafficked areas. When building a house next to a heavily trafficked road, the windows should be noise isolated, and the house protected from air pollution.

Ridesharing or carpooling is a popular research topic [13]. The sharing culture is up and coming and a great measure to save money and reduce climate change and congestion. There are test runs on trying to get people driving to and from approximately the same locations around the same time to drive together. The traffic and the greenhouse gas emissions will be reduced. Trajectories will have to be compared to determine who should share a ride. There is some research in this field already. The most straightforward is to compare start and end coordinates to find matches [18, 13]. However, comparing entire trajectories opens up for people

only driving partly together. A passenger can be let off before the driver arrives at their destination or picked up after they have left their place of departure. For this purpose, entire trajectories must be compared to find a high percentage of overlapping in the routes.

In Section 2.1, we introduced the concept of map-matching and clustering based on trajectory similarity. Kim and Mahmassani used the Longest Common Subsequence algorithm to match trajectories for clustering purposes [26]. The LCSS algorithm can easily be switched out with any three algorithms from this study. However, they used this algorithm to find their overall similarity by focusing on entire tracks, not sub-tracks. They calculate the percentage of overlap between the tracks to measure their similarity. The resulting number is the baseline for the clustering. The algorithms from this thesis can likely also be used for that purpose, but it has not been explored yet. Therefore, it is difficult to say whether they would be a better or worse choice for this problem.

7. Conclusion and Further Work

7.1. Conclusion

This thesis has explored three sequence alignment algorithms and their application to the trajectory similarity problem. The similarity calculations have contributed to identifying the most trafficked roads in a city. The experiments have explored the different traits of the three algorithms, both for identifying the most trafficked road and trajectory similarity in general.

7.2. Evaluation

The overall results found were very similar. The algorithms only differed in some details. Smith-Waterman was overall more accurate in detecting matches. Both Needleman-Wunsch and Dynamic Time Warping risked getting lost in the backtracking step. These algorithms will probably perform better on more similar input sequences. Dynamic Time Warping found many more matching points than the two others. It can be argued that DTW found these due to not stopping the backtracking in time.

Overall, all algorithms performed well. They could find the most trafficked roads and generated a very similar result. The question of which algorithm best suits this problem does not have a clear answer. DTW generated higher frequencies, more coordinate points, and longer tracks. It is difficult to determine if that represents reality.

Due to the limitations caused by creating the result list, the results generated from the isolated algorithms are more valid than the complete result lists. All algorithms have their limitations. They all might include coordinate points that should not be there. They also fail to include points that should. Not including essential alignments can have more considerable consequences than including some

alignments that should not be there. Wrongly included alignments are usually no longer than four or five coordinate points. They will not affect the final result a great deal. For this reason, as well as the unstable backtracking of DTW and NW, it would be preferable to use Smith-Waterman for this problem. That is mainly because Smith-Waterman is better suited to output sub-trajectories. As a bonus, it is also easier to work with and faster than the other algorithms.

7.3. Contributions

The main contribution of this project is how the three sequence alignment algorithms are used for a new purpose. It has been proved that they can solve the trajectory similarity problem. Especially Smith-Waterman is valuable here. As far as the author knows, Smith-Waterman has not been scientifically used on trajectories before. This algorithm performed nicely and is well suited for this problem.

This thesis has also introduced a new way of mapping traffic. Identifying the most pressured roads in the city has not been done in this way before. It opens up new ways of analyzing traffic data. Much previous work focuses solely on congestion formation, not the number of cars. This study opens up to solving other problems, like mapping air and noise pollution and knowing people's transportation behavior.

7.4. Limitations and Further Work

As this project explores a new field, the method has many limitations. There are steps performed outside of the three alignment algorithms that are not to be entirely trusted. Many decisions taken throughout the project affect each other and the result very much. Thus, it could have significant consequences if one is not perfected. This dependence is a considerable limitation as it is challenging to know which step is the weakest. These limitations include how the result list is made. This step has to be researched more extensively to find the best method for gathering matching alignments. How the tracks are divided into sub-tracks is also a step that can be researched more thoroughly. Finally, the formula for calculating the score of the result tracks is highly dependent on the data. It is not sure whether it would work for other data or whether it is the best method for this data.

As an alternative to creating a result list, an exciting experiment could be to run the alignment algorithms multiple times. The output from the algorithms' first run will be used as input for another run of the algorithms. It would combine many overlapping result tracks explained in Section 5.1. The final result list will not include the score deciding their ordering but will include fewer and more accurate result tracks. The less frequent roads would be filtered out, and only the most frequently overlapping tracks will be left.

The most noteworthy limitation of using the sequence algorithms for this purpose is the percentage of similarity. That is a number explaining how much of the data is overlapping. As explained, the percentage of similarity Smith-Waterman and Needleman-Wunsch is made for is 99,4 % [50], which is far away from the similarity percentage in the dataset used in this study. It is a disadvantage of using these algorithms and could be why NW fails the backtracking step in some cases.

Another limitation of the sequence alignment algorithms could be directionality. It depends on the application of the method. The direction of the cars' movement is

valuable information when mapping traffic and identifying congestion. However, when identifying noise and air pollution, it does not matter which direction the cars are driving.

Throughout the research for this project, the Longest Common Subsequence algorithm has been prominent. It became clear that this algorithm has been used as much as Dynamic Time Warping for trajectory similarity. A study conducted by Keshk et al. compared the LCSS algorithm to Needleman-Wunsch and Smith-Waterman [24]. However, they only considered time complexity and concluded LCSS was the fastest of the three. It would be interesting to compare this algorithm to the three tested out in this thesis and explore how it would perform.

The tracks in the dataset were not very accurate. Intersections were skipped, and sometimes the GPS got lost. That is a limitation of GPS as a tracking system. Something that could help the result be more accurate is to compare them to the road network instead of comparing them to each other. That is what we call map matching. It would help the results, but it requires a road network for the comparison. Hence, it is a more costly operation.

Another dataset would not help the accuracy as the imprecision primarily lies with the GPS. Nonetheless, it could verify the method if it were to work on different data, as the methods explained in this thesis are highly adapted to the dataset. Preferably, it would be a larger city with wider roads to see how well the algorithm performs on data from another type of city. This project is also based on taxi data, which does not necessarily represent the daily traffic behavior.

An expansion of the data could provide other information. Much previous work includes the time of day and week for their analysis. It provides much information about the transportation behavior of the people. It shows where the traffic is most pressing throughout the day and the differences between weekdays and weekends. Including this information will complicate the process and the algorithms, and the entire process will have to be adapted. However, this step's amount of additional information could make it worthwhile.

8. Bibliography

[1] "Taxi Service Trajectory - Prediction Challenge - Dataset by UCI," *data.world*. https://data.world/uci/taxi-service-trajectory-prediction-challenge-ecml-pkdd.

[2] "Smith–Waterman Algorithm," *Wikipedia*, Dec. 2021, Accessed: Dec. 10, 2021. [Online].

[3] "Needleman–Wunsch Algorithm," *Wikipedia*, Nov. 2021, Accessed: Nov. 21, 2021. [Online].

[4] S. Abraham and P. Sojan Lal, "Spatio-Temporal Similarity of Network-Constrained Moving Object Trajectories Using Sequence Alignment of Travel Locations," *Transportation Research Part C: Emerging Technologies*, vol. 23, pp. 109–123, Aug. 2012, doi: 10.1016/j.trc.2011.12.008.

[5] G. Al-Naymat, S. Chawla, and J. Taheri, "SparseDTW: A Novel Approach to Speed up Dynamic Time Warping," no. arXiv:1201.2969. arXiv, Jan. 2012.

[6] V. Albino, U. Berardi, and R. M. Dangelico, "Smart Cities: Definitions, Dimensions, Performance, and Initiatives," *Journal of Urban Technology*, vol. 22, no. 1, pp. 3–21, Jan. 2015, doi: 10.1080/10630732.2014.942092.

[7] D. R. Aleko and S. Djahel, "An IoT Enabled Traffic Light Controllers Synchronization Method for Road Traffic Congestion Mitigation," in *2019 IEEE International Smart Cities Conference (ISC2)*, Oct. 2019, pp. 709–715, doi: 10.1109/ISC246665.2019.9071667.

[8] M. Batty, "Smart Cities, Big Data," *Environment and Planning B: Planning and Design*, vol. 39, no. 2, pp. 191–193, Apr. 2012, doi: 10.1068/b3902ed.

[9] D. J. Berndt and J. Clifford, "Using Dynamic Time Warping to Find Patterns in Time Series," *AAAI Technical Report WS*, no. 3, pp. 359–370, Apr. 1994.

[10] D. J. Briggs *et al.*, "A Regression-Based Method for Mapping Traffic-Related Air Pollution: Application and Testing in Four Contrasting Urban Environments," *Science of The Total Environment*, vol. 253, no. 1, pp. 151–167, May 2000, doi: 10.1016/S0048-9697(00)00429-0.

[11] L. Chen, M. T. Özsu, and V. Oria, "Robust and Fast Similarity Search for Moving Object Trajectories," in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, Jun. 2005, pp. 491–502, doi: 10.1145/1066157.1066213.

[12] F. Cristino, S. Mathôt, J. Theeuwes, and I. Gilchrist, "ScanMatch: A Novel Method for Comparing Fixation Sequences," *Behavior research methods*, vol. 42, pp. 692–700, Aug. 2010, doi: 10.3758/BRM.42.3.692.

[13] M. Cruz, H. T. Macedo, and A. P. Guimarães, "Grouping Similar Trajectories for Carpooling Purposes," *2015 Brazilian Conference on Intelligent Systems (BRACIS)*, 2015, doi: 10.1109/BRACIS.2015.36.

[14] European Commission, "Sustainable Transport." https://transport.ec.europa.eu/transport-themes/sustainable-transport_en, Accessed: May 25, 2022. [Online].

[15] European Commission, "Smart Cities." https://ec.europa.eu/info/eu-regional-and-urban-development/topics/cities-and-urban-development/city-initiatives/smart-cities_en, Accessed: May 13, 2022. [Online].

[16] J. Gudmundsson, P. Laube, and T. Wolle, "Computational Movement Analysis," in *Springer Handbook of Geographic Information*, Berlin, Heidelberg: Springer, 2012, pp. 423–438.

[17] T. Güyer, B. Atasoy, and S. Somyürek, "Measuring Disorientation Based on the Needleman-Wunsch Algorithm," *International Review of Research in Open and Distributed Learning*, vol. 16, no. 2, pp. 188–205, 2015, doi: 10.19173/irrodl.v16i2.2016.

[18] Z. Hong, Y. Chen, H. S. Mahmassani, and S. Xu, "Commuter Ride-Sharing Using Topology-Based Vehicle Trajectory Clustering: Methodology, Application and Impact Evaluation," *Transportation Research Part C: Emerging Technologies*, vol. 85, pp. 573–590, Dec. 2017, doi: 10.1016/j.trc.2017.10.020.

[19] J.-R. Hwang, H.-Y. Kang, and K.-J. Li, "Spatio-Temporal Similarity Analysis Between Trajectories on Road Networks," in *Perspectives in Conceptual Modeling*, vol. 3770, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 280–289.

[20] J.-R. Hwang, H.-Y. Kang, and K.-J. Li, "Searching for Similar Trajectories on Road Networks Using Spatio-temporal Similarity," in *Advances in Databases and Information Systems*, vol. 4152, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–295.

[21] E. Hyytiä and J. Virtamo, "On Traffic Load Distribution and Load Balancing in Dense Wireless Multihop Networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2007, p. 15, Dec. 2007, doi: 10.1155/2007/16932.

[22] Y. Jararweh, M. Al-Ayyoub, M. Fakirah, L. Alawneh, and B. B. Gupta, "Improving the Performance of the Needleman-Wunsch Algorithm Using Parallelization and Vectorization Techniques," *Multimedia Tools and Applications*, vol. 78, no. 4, pp. 3961–3977, Feb. 2019, doi: 10.1007/s11042-017-5092-0.

[23] S. Kamran and O. Haas, "A Multilevel Traffic Incidents Detection Approach: Identifying Traffic Patterns and Vehicle Behaviours Using Real-Time GPS Data," in *2007 IEEE Intelligent Vehicles Symposium*, Jun. 2007, pp. 912–917, doi: 10.1109/IVS.2007.4290233.

[24] A. Keshk, M. Ossman, and L. Fathi Hussein, "Fast Longest Common Subsequences for Bioinformatics Dynamic Programming," *International Journal of Computer Applications*, vol. 57, no. 22, pp. 12–18, Nov. 2012, doi: 10.5120/9419-3569.

[25] A. Khedher, I. Jraidi, and C. Frasson, "Local Sequence Alignment for Scan Path Similarity Assessment," *International Journal of Information and Education Technology*, vol. 8, pp. 482–490, Jan. 2018, doi: 10.18178/ijiet.2018.8.7.1086.

[26] J. Kim and H. S. Mahmassani, "Spatial and Temporal Characterization of Travel Patterns in a Traffic Network Using Vehicle Trajectories," *Transportation Research Procedia*, vol. 9, pp. 164–184, Jan. 2015, doi: 10.1016/j.trpro.2015.07.010.

[27] M. Kohan and J. M. Ale, "Discovering Traffic Congestion through Traffic Flow Patterns Generated by Moving Object Trajectories," *Computers, Environment and Urban Systems*, vol. 80, p. 101426, Mar. 2020, doi: 10.1016/j.compenvurbsys.2019.101426.

[28] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory Clustering: A Partition-and-Group Framework," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data - SIGMOD '07*, Beijing, China, 2007, p. 593, doi: 10.1145/1247480.1247546.

[29] T. Li, J. Wu, A. Dang, L. Liao, and M. Xu, "Emission Pattern Mining Based on Taxi Trajectory Data in Beijing," *Journal of Cleaner Production*, vol. 206, pp. 688–700, Jan. 2019, doi: 10.1016/j.jclepro.2018.09.051.

[30] A. Mishra, "Time Series Similarity Using Dynamic Time Warping -Explained," *Walmart Global Tech Blog*. https://medium.com/walmartglobaltech/time-series-similarity-

using-dynamic-time-warping-explained-9d09119e48ec, Dec. 2020, Accessed: May 20, 2022. [Online].

[31] M. Müller, *Information Retrieval for Music and Motion*. New York: Springer, 2007.

[32] S. B. Needleman and C. D. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, Mar. 1970, doi: 10.1016/0022-2836(70)90057-4.

[33] D. Okada, F. Ino, and K. Hagihara, "Accelerating the Smith-Waterman Algorithm with Interpair Pruning and Band Optimization for the All-Pairs Comparison of Base Sequences," *BMC Bioinformatics*, vol. 16, no. 1, p. 321, Oct. 2015, doi: 10.1186/s12859-015-0744-4.

[34] X. Olive and L. Basora, "Identifying Anomalies in Past En-Route Trajectories with Clustering and Anomaly Detection Methods," Jun. 2019.

[35] F. Pinna, F. Masala, and C. Garau, "Urban Policies and Mobility Trends in Italian Smart Cities," *Sustainability*, vol. 9, no. 4, p. 494, Apr. 2017, doi: 10.3390/su9040494.

[36] L. Popa, A. Rostamizadeh, R. Karp, C. Papadimitriou, and I. Stoica, "Balancing Traffic Load in Wireless Networks with Curveball Routing," in *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, New York, NY, USA, Sep. 2007, pp. 170–179, doi: 10.1145/1288107.1288131.

[37] J. Qin, G. Mei, and L. Xiao, "Building the Traffic Flow Network with Taxi GPS Trajectories and Its Application to Identify Urban Congestion Areas for Traffic Planning," *Sustainability*, vol. 13, no. 1, p. 266, Jan. 2021, doi: 10.3390/su13010266.

[38] Y. Qing, Z. Yue, R. Chen, Z. Jingwei, X. Hu, and Y. Zhou, "Real-Time Detection of Traffic Congestion Based on Trajectory Data," *The Journal of Engineering*, vol. 2019, Jul. 2019, doi: 10.1049/joe.2019.0872.

[39] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Current Map-Matching Algorithms for Transport Applications: State-of-the Art and Future Research Directions," *Transportation Research Part C: Emerging Technologies*, vol. 15, no. 5, pp. 312–328, Oct. 2007, doi: 10.1016/j.trc.2007.05.002.

[40] L. R. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, N.J: PTR Prentice Hall, 1993.

[41] C. Ratanamahatana and E. Keogh, "Everything You Know about Dynamic Time Warping Is Wrong," Jan. 2004.

[42] I. H. Sarker, "Smart City Data Science: Towards Data-Driven Smart Cities with Open Research Issues," *Internet of Things*, vol. 19, Aug. 2022, doi: 10.1016/j.iot.2022.100528.

[43] M. Sharif and A. A. Alesheikh, "Similarity Measurement of Trajectories Based on Contextual Data in Constrained Euclidean Space," *Journal of Geomatics Science and Technology*, vol. 5, no. 4, pp. 113–125, Jun. 2016.

[44] K. Sheng, Z. Liu, D. Zhou, A. He, and C. Feng, "Research on Ship Classification Based on Trajectory Features," *The Journal of Navigation*, vol. 71, no. 1, pp. 100–116, Jan. 2018, doi: 10.1017/S0373463317000546.

[45] Q. Shi, K. Zhang, J. Weng, Y. Dong, S. Ma, and M. Zhang, "Evaluation Model of Bus Routes Optimization Scheme Based on Multi-Source Bus Data," *Transportation Research Interdisciplinary Perspectives*, vol. 10, Jun. 2021, doi: 10.1016/j.trip.2021.100342.

[46] D. Silva, R. Giusti, E. Keogh, and G. Batista, "Speeding up Similarity Search under Dynamic Time Warping by Pruning Unpromising Alignments," *Data Mining and Knowledge Discovery*, vol. 32, Jul. 2018, doi: 10.1007/s10618-018-0557-y.

[47] T. F. Smith and M. S. Waterman, "Identification of Common Molecular Subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, Mar. 1981, doi: 10.1016/0022-2836(81)90087-5.

[48] F. K. Soong and A. E. Rosenberg, "On the Use of Instantaneous and Transitional Spectral Information in Speaker Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 6, pp. 871–879, Jun. 1988, doi: 10.1109/29.1598.

[49] T. Stenovec, "Google Has Gotten Incredibly Good at Predicting Traffic — Here's How," *Business Insider*. https://www.businessinsider.com/how-google-maps-knows-about-traffic-2015-11, Accessed: May 07, 2022. [Online].

[50] The 1000 Genomes Project Consortium *et al.*, "A Global Reference for Human Genetic Variation," *Nature*, vol. 526, no. 7571, pp. 68–74, Oct. 2015, doi: 10.1038/nature15393.

[51] K. Toohey and M. Duckham, "Trajectory Similarity Measures," *SIGSPATIAL Special*, vol. 7, no. 1, pp. 43–50, May 2015, doi: 10.1145/2782759.2782767.

[52] United Nations, Department of Economic and Social Affairs, and Population Division, *World Urbanization Prospects: The 2007 Revision Population Database*. 2008.

[53] United Nations, Department of Economic and Social Affairs, and Population Division, *World Urbanization Prospects: The 2018 Revision*. 2019.

[54] M. Vlachos, D. Gunopulos, and G. Kollios, "Robust Similarity Measures for Mobile Object Trajectories," in *Proceedings. 13th International Workshop on Database and Expert Systems Applications*, Sep. 2002, pp. 721–726, doi: 10.1109/DEXA.2002.1045983.

[55] Y. Wakuda, S. Asano, N. Koshizuka, and K. Sakamura, "An Adaptive Map-Matching Based on Dynamic Time Warping for Pedestrian Positioning Using Network Map," in *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, Apr. 2012, pp. 590–597, doi: 10.1109/PLANS.2012.6236932.

[56] Z. Wang, T. Novack, Y. Yan, and A. Zipf, "Quiet Route Planning for Pedestrians in Traffic Noise Polluted Environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 12, pp. 7573–7584, Dec. 2021, doi: 10.1109/TITS.2020.3004660.

[57] Z. Wang, M. Lu, X. Yuan, J. Zhang, and H. Van De Wetering, "Visual Traffic Jam Analysis Based on Trajectory Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2159–2168, Dec. 2013, doi: 10.1109/TVCG.2013.228.

[58] W. Wen, "An Intelligent Traffic Management Expert System with RFID Technology," *Expert Systems with Applications*, vol. 37, no. 4, pp. 3024–3035, Apr. 2010, doi: 10.1016/j.eswa.2009.09.030.

[59] M. Wiltgen, "Algorithms for Structure Comparison and Analysis: Homology Modelling of Proteins," in *Encyclopedia of Bioinformatics and Computational Biology*, Oxford: Academic Press, 2019, pp. 38–61.

[60] J. Zhou, X. Ruan, X. Shi, and C. C. Caprani, "An Efficient Approach for Traffic Load Modelling of Long Span Bridges," *Structure and Infrastructure Engineering*, vol. 15, no. 5, pp. 569–581, May 2019, doi: 10.1080/15732479.2018.1555264.

[61] M. Zhou, Z. Tian, K. Xu, H. Wu, Q. Pu, and X. Yu, "Construction of Time-Stamped Mobility Map for Path Tracking via Smith-Waterman Measurement Matching," *Mathematical Problems in Engineering*, vol. 2014, pp. 1–17, Mar. 2014, doi: 10.1155/2014/673159.

[62] M. Čavojský, M. Drozda, and Z. Balogh, "Analysis and Experimental Evaluation of the Needleman-Wunsch Algorithm for Trajectory Comparison," *Expert Systems with Applications*, vol. 165, p. 114068, Mar. 2021, doi: 10.1016/j.eswa.2020.114068.

Appendix A: Appendix

A.1. Code

The link to the GitHub repository is the following

https://github.com/kajasl/ideal-journey

The project was implemented in Jupyter Notebook. The complete result lists is also included in the GitHub repository.

A.2. Complete List of Matching Tracks to Track 2

Table 4. The table includes an overview of all tracks the three algorithms found to be matching track 2 in the dataset.

Smith-Waterman	**Needleman-Wunsch**	**Dynamic Time Warping**
T18	T18	
T32	T32	T32
T33	T33	T33
	T51671577	
T109	T109	T109

Smith-Waterman	Needleman-Wunsch	Dynamic Time Warping
T110	T110	T110
T112	T112	T112
T130	T130	T130
T135	T135	T135
T136	T136	T136
T147	T147	T147
T161674001		
T166	T166	T166
		T18014434
T190		
T194		
T202	T202	T202
T207675434		

Smith-Waterman	Needleman-Wunsch	Dynamic Time Warping
T218	T218	T218
T221		
T22716476	T22716476	T22716476
T230	T230	T230
T23116727	T23116727	T23116727
T236	T236	T236
T238	T238	T238
T242676972	T242676972	T242676972
T254	T254	T254
		T258
T263677479	T263677479	T263677479
T264	T264	T264
T267	T267	T267

Smith-Waterman	Needleman-Wunsch	Dynamic Time Warping
T307	T307	T307

A.3. Result Map Only Based on Frequency

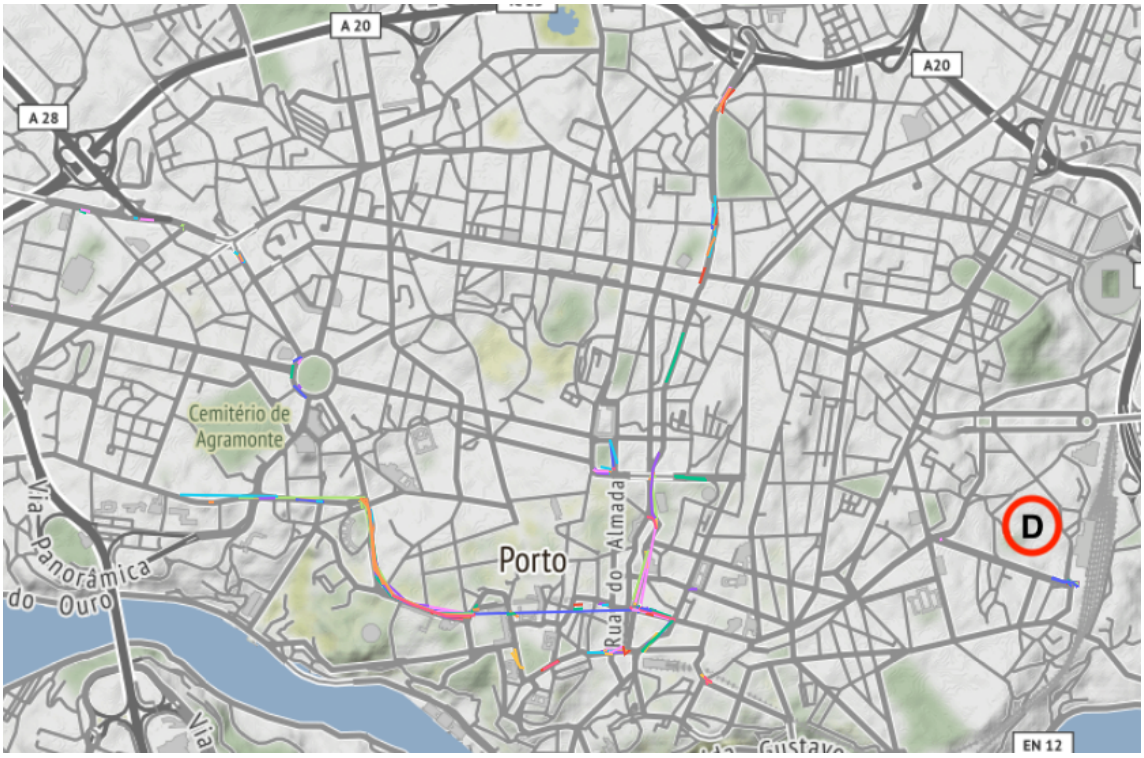


Figure 21. A map created from the Smith-Waterman algorithm where only the frequency of tracks counts for the value deciding their order and if they are included in the list. This is the complete result list. There are tracks where many points in between have been removed. It is unclear why this happened.

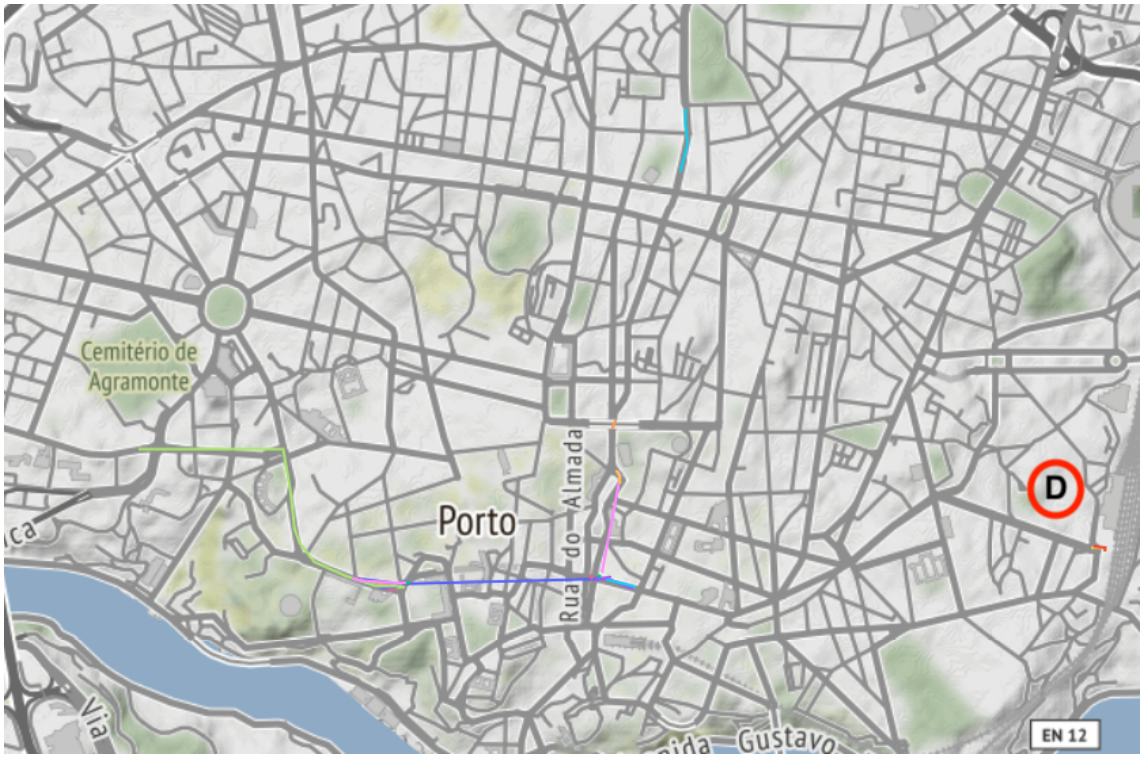
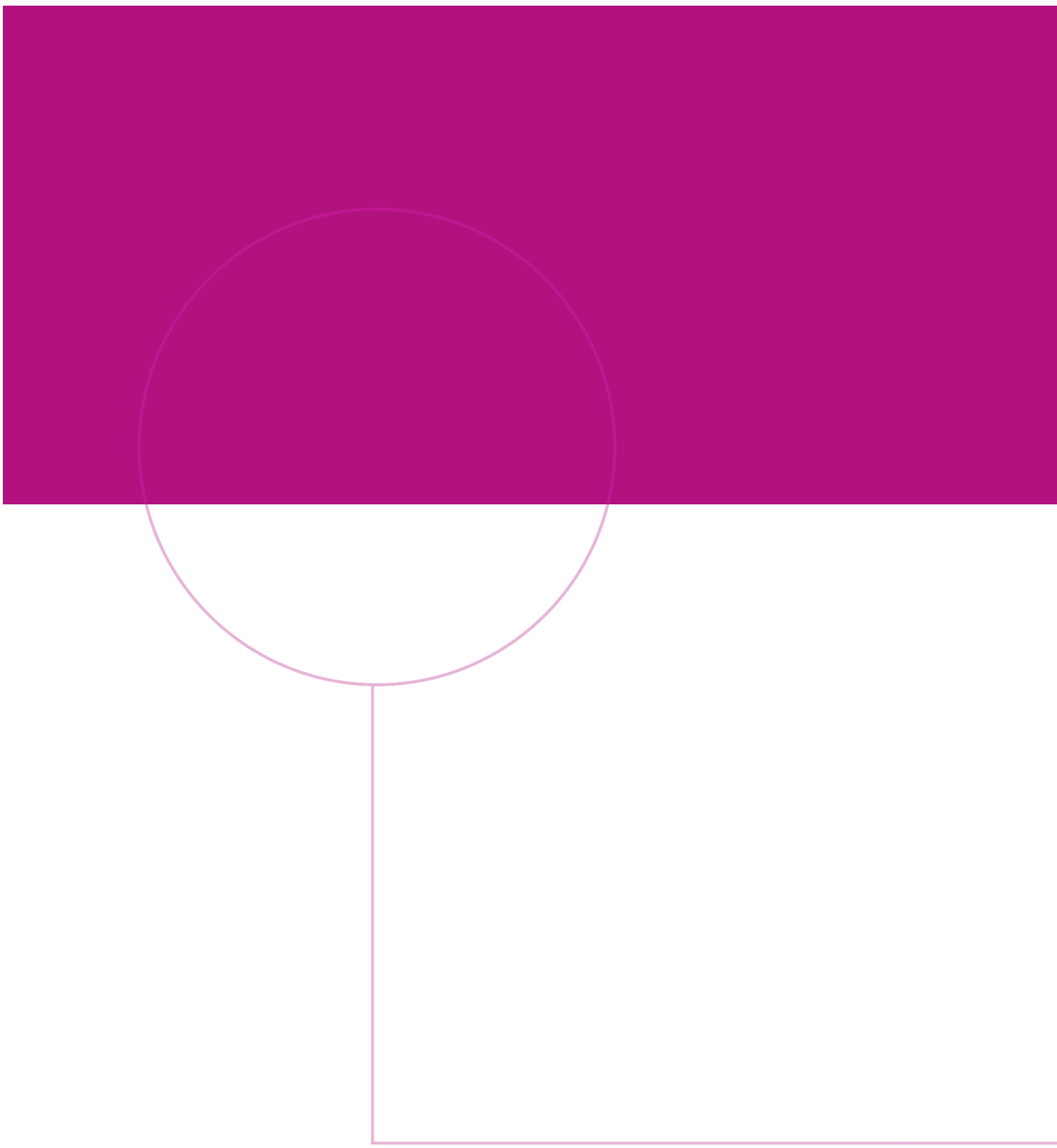


Figure 22. This is the top 20 list created from the same result as Figure 21



NTNU

Norwegian University of
Science and Technology