Documentation Group 03 - Assignment 2

Moritz Brandl: a1408112, Lukas Miklautz: a01246128 and Raphael Mitsch: a1006529

University of Vienna

Abstract. In this paper the results for the second assignment of the Data Mining course are documented and discussed. We found that DTW leads to significant improvements over Euclidean distances for the individual trips. Extracting features from the time series and then comparing them using the Euclidean distance leads to models of similar quality. We conclude with suggestions for improvements and further ideas we weren't able to pursue due to lack of time.

Keywords: data mining, mobility data, transport modes, time series clustering

1 Introduction

The second assignment of the Data Mining course was about clustering of mobility data. The data was recorded by each student separately and aggregated for each group. Our group chose to record walking, tram and metro trips. To record the data we had to install an android app on our smart phones, which if activated recorded the available sensor data of the device. During the whole project we used the accelerometer data for clustering and GPS data for plotting the trips on a map. Location data was used for illustrative purposes only.

Note that all of our code is written in Python, including the preprocessing routines. We didn't use the supplied R code and used for guidance only. This was due to the reason that we did not want to introduce too many dependencies for the project. This resulted in an extra effort, but we think this decision has paid off in the longer term. Our code for this project can be found on github[1].

2 Data Collection

In this section the data gathering process is discussed. We had the following tokens and devices with which we recorded the data:

Name	MatrNr	Device	Token
Moritz Brandl	a1408112	Samsung Galaxy S5 mini	355007075245007
Lukas Miklautz	a01246128	One Plus E1001	868049020858898
Raphael Mitsch	a1006529	LG-E460	358568053229914

[1] https://github.com/univie-datamining-team3/assignment2

We recorded the three modes WALK, METRO and TRAM. We faced some issues during the recording:

- We had a varying size of trips for equal recording length. E.g. Raphael's phone recorded only accelerometer and GPS data, so his trips were in the kilobyte range, while the recordings of Moritz and Lukas were in the megabyte range.
- Sometimes the sensors on the phone stopped recording for a few seconds, which resulted in gaps in the recorded data.
- The app crashed sometimes during data collection, apparently deleting a currently running trip dataset and sometimes also already persisted trips. This didn't happen often though.

3 Preprocessing

Our preprocessing pipeline executes the following steps (in this sequence):

1. Download of all available datasets, extracting individual files for trips.
2. Remove trips without annotations (since we can't infer which transport mode was used).
3. Replace non-existent parts of the dataset (e. g. location data, if no GPS signal available) with empty dataframes.
4. Discard trips lasting less than 10 minutes.
5. Cut off a trip's first and last 60 seconds.
6. Calculate piecewise aggregate approximation.
7. Cut trips in snippets of 30 seconds.
8. Calculate the distance matrix for all combinations of trips applying the defined distance metric on either the column *total* or the individual columns x, y, z.[2]

We persisted the preprocessing results as well as the calculated distance matrices on disk and implemented procedures to load the data into memory to enable a faster work flow.

4 Visualization and Exploratory Data Analysis

This section explores the data we gathered and visualizes some of the findings we discovered. First of all we will look at part of a table summarizing each trip as shown in fig.1. The time column represents the total trip time. This frame shows the already preprocessed data.

Fig.2 gives information about the number of trips per mode type each of our group members has collected and splits it into scripted and ordinary trips given the note information of the trips.

[2] Note that our implementation supports a variety of distance metrics if *total* is used as defining criterion, but only the Euclidean distance if the individual columns are chosen.

	time	mode	notes	user	Start	Stop	trip_length
0	2017-12-11 12:48:06.493	METRO	scripted	Moritz	2017-12-11 12:48:07.665	2017-12-11 12:59:59.683	00:11:52.018000
1	2017-12-13 13:08:50.066	METRO	scripted	Raphael	2017-12-13 13:08:50.302	2017-12-13 13:23:14.050	00:14:23.748000
2	2017-12-09 20:04:36.073	METRO	scripted	Raphael	2017-12-09 20:04:36.335	2017-12-09 20:15:55.682	00:11:19.347000
3	2017-12-08 14:48:43.804	METRO	scripted	Raphael	2017-12-08 14:48:44.143	2017-12-08 14:59:39.862	00:10:55.719000
4	2017-12-01 12:02:52.760	METRO	ordinary u2praterster_mq	Lukas	2017-12-01 12:02:52.981	2017-12-01 12:13:14.779	00:10:21.798000

Fig. 1. Trip summary

			count
user	**mode**	**type**	
Lukas	**METRO**	**ordinary**	5
		scripted	6
	TRAM	**ordinary**	5
		scripted	5
	WALK	**ordinary**	9
		scripted	5
Moritz	**METRO**	**ordinary**	5
		scripted	5
	TRAM	**ordinary**	6
		scripted	4
	WALK	**ordinary**	6
		scripted	5
Raphael	**METRO**	**ordinary**	6
		scripted	5
	TRAM	**ordinary**	7
		scripted	7
	WALK	**ordinary**	5
		scripted	5

Fig. 2. Trip collection summary

In order to get a better understanding of the data we looked at the android documentation for the sensor data provided. Acceleration data is measured in $\frac{m}{s^2}$ and defines an object's change in velocity over time. The raw acceleration data contains the true acceleration of the device plus gravitational acceleration and offset. To generate clean data it would be necessary to remove the acceleration force due to gravitation as well as the offset [1].

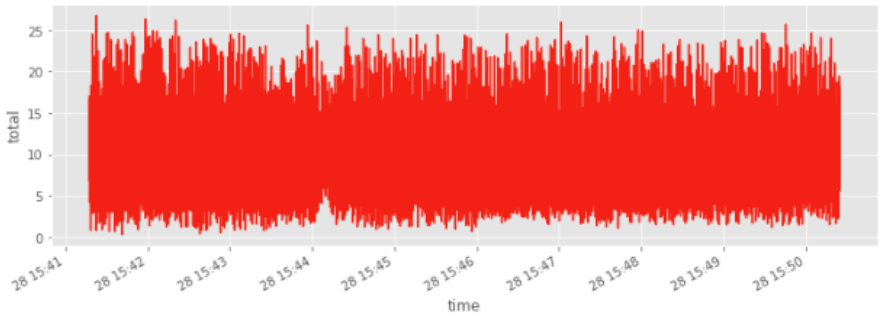

Fig. 3. Acceleration pattern of a walk

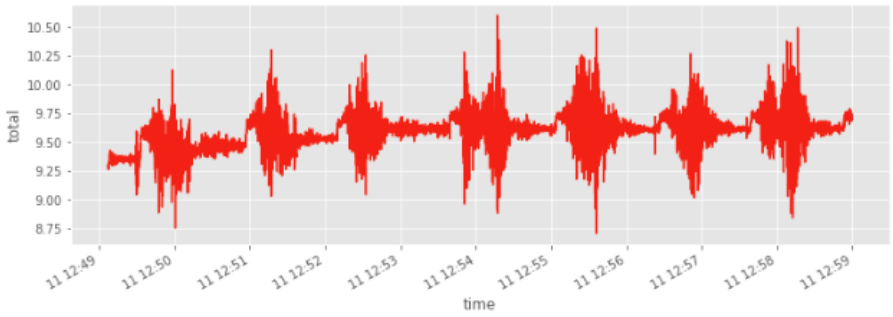

Fig. 4. Acceleration pattern of a metro trip

Figures 3, 4 and 5 show the total acceleration data for each mode of transport. Each of those recordings was recorded by a different group member. By solely looking at fig. 3 it is hard to see a pattern. The data seems to be evenly spaced in both directions of the amplitude. In comparison there are distinct patterns visible when looking at fig. 4 and 5. For these modes we can see how the transport

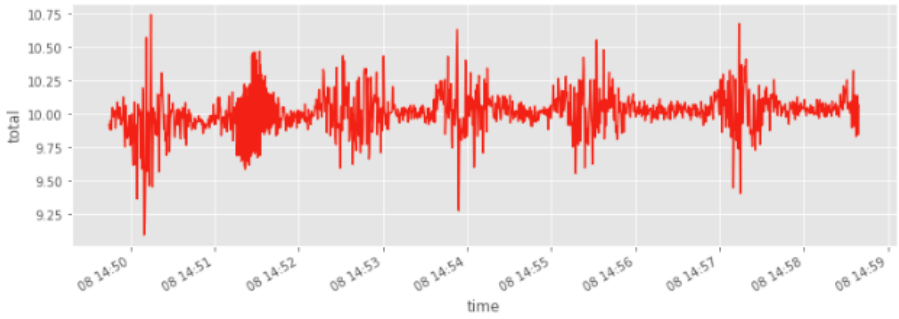

Fig. 5. Acceleration pattern of a tram trip

vehicle accelerates and breaks between its stops. As mentioned before the data is neither cleaned of gravitation nor offset, but nonetheless we can see a general pattern.

Most of our trips lasted for pretty much exactly 10 minutes, one outlier is a trip of 30 minutes - see Figure 6. After some investigation we concluded that this trip seems to be an erroneously recorded one. We decided not to update code and documentation due to our assumption that this trip, since it was recorded as WALK and WALK trips are generally easily separable from other transport modes, most likely doesn't distort our result significantly. The total amount of minutes spent (without the outlier) is about 19 hours and 30 minutes.

For a spatial overview of our trips see Figure 7.

Visualization with TSNE

Fig. 9 shows that trips with the mode "WALK" are distinct from "TRAM" and "METRO". Trips with "TRAM" and "METRO" do overlap, which indicates possible problems for classification/clustering. We calculated the euclidean distance of the euclidean norm of the x,y,z accelerometer sensor data. Each point in the distance matrix in fig. 8 is the distance of one trip segment to another one and each row of the distance matrix corresponds to the trips segment distances to all other trip segments. We chose the TSNE plot as it allows us to plot our high dimensional data in a 2-dimensional scatter plot and is designed to group similar objects. We also utilized an implementation of Bayesian optimization to find the optimal low-dimensional representation in terms of retaining points' neighborhood, although our optimization process didn't yield improvements to parameters picked by hand (probably we didn't choose the optimization criterion appropriately - we would have liked to update the criteria, but didn't have the time to do so).

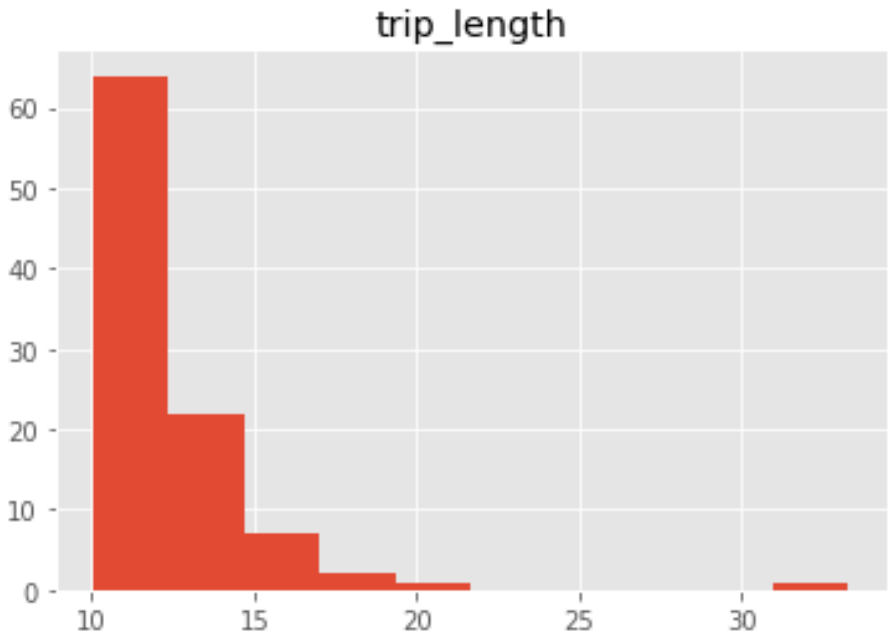

Fig. 6. Histogram of trip lengths.

Fig. 7. Heat map of all recorded trips.

	distance_0	distance_1	distance_2	distance_3	distance_4	distance_5	distance_6	distance_7	distance_8	dist
0	0.000000	5.657964	2.996962	4.485049	6.819119	4.090885	8.093241	6.275309	6.153399	8.
1	5.657964	0.000000	4.887508	5.638583	6.413127	4.452474	7.312200	5.687715	5.497745	7.
2	2.996962	4.887508	0.000000	3.728035	5.850668	2.839858	6.877979	4.857203	4.798214	6.
3	4.485049	5.638583	3.728035	0.000000	5.716500	2.527824	5.260878	3.450875	3.150206	5.
4	6.819119	6.413127	5.850668	5.716500	0.000000	4.544327	6.224123	4.904976	4.664025	6.

Fig. 8. Euclidean distance data used for TSNE plot in figure 9

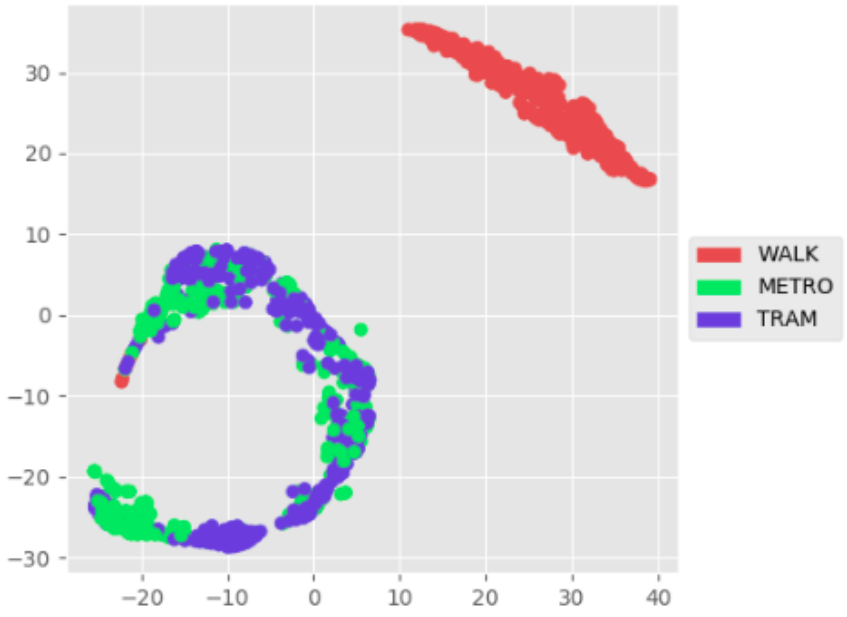

Fig. 9. TSNE plot of scripted trips with perplexity of 50 and a learning rate of 1000. This plot shows only the scripted trips which have been cut in 30 second snippets and transformed to a euclidean distance matrix.

5 Investigated Distance Functions

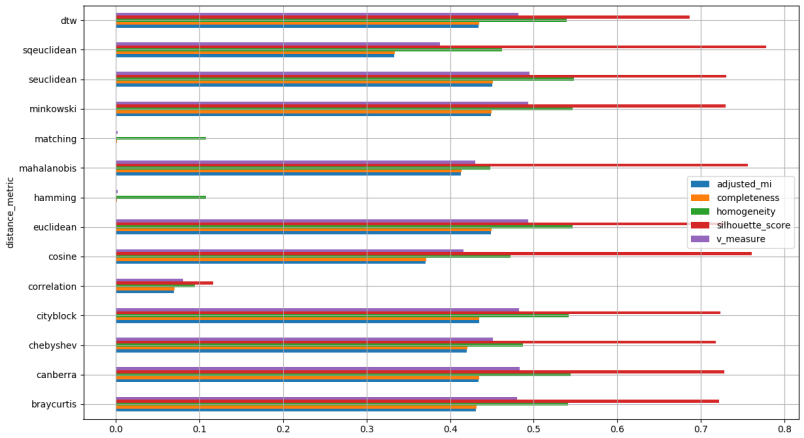

Fig. 10. Comparison of distance measures supported by sklearn and scipy's cdist function used for calculating the distance matrix and clustered with k-means. All scores, except the silhouette score have been evaluated against the transport modes.

We evaluated all distance functions for discrete data supported by sklearn as well as DTW[3] and evaluated them by passing on the corresponding distance matrix between all trip snippets to k-means and calculating various clustering objectives. While some yielded inferior outcomes, most of them generated results of comparable quality (see figure 10).

Hence we used either the L2 norm or DTW when calculating distances between trips and the L2 norm when calculating distances between sets of engineered features. We also did an informal evaluation of the effect of various distance measures for calculating similarities between engineered features, which led to similar results.

[3] We used this fastDTW implementation.

5.1 Calculation time

All distance functions for the n2 column - except DTW and Mahalanobis - terminated within less than a minute for all trip segments[4]. The Euclidean for the individual columns took a few minutes with the mentioned setup.

6 Feature Engineering

Our attempts at engineering features from the collected time series are described below. While they yielded some interesting insights, none of them improved the resulting model significantly.

6.1 Combining different distance measures

In this approach we calculated three different distance measures from the original table with the trips cut in 30 second snippets. The underlying idea with this was to get different "views" of the data. From each of those distance matrices we calculated basic features like quantiles, standard deviation and mean. This new feature matrices are highly correlated, because they share the same underlying data. In order to decorrelate them we used PCA on the combined features and extracted 4 principle components, which explain about 90% of the variance. The new feature space allows us to inspect new clusters as visualized in figure 11, compare this to figure 9 where only the euclidean distances were used. We can see e.g. that the blue token has separated clusters for TRAM/METRO and WALK.

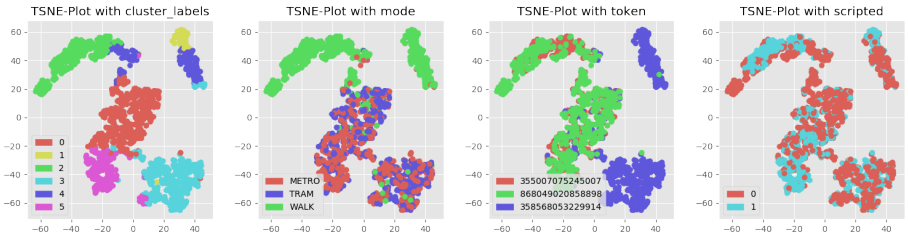

Fig. 11. TSNE-Plot of engineered features from distance matrices calculated by euclidean, cosine and correlation distance. Each plot shows a different color encoding. The first plot on the left shows a clustering performed by HDBSCAN, for details see this notebook

.

[4] With installed Cython and numpy linked to OpenBLAS. Calculating the distances without these dependencies might lead to calculation times higher by a factor of tens or hundreds.

6.2 Top speed

We intended to reconstruct each trip snippet's average speed from the collected acceleration data. Our intuition was that since tram and metro trips are hard to distinguish, average speed would be a feature that might be distinctive enough to separate them (since metros reach higher typically speed than trams). As already mentioned earlier, acceleration can be converted to velocity. Some devices also provide linear acceleration data which according to the documentation is cleansed of gravitational acceleration. Unfortunately, after some research we dropped the idea of calculating the velocity. To be able to calculate the velocity, we would have needed to know the devices orientation at every point in time. And as our research concluded the integration of acceleration data to receive velocity or position is especially error prone given the inaccuracy of the devices cheap acceleration sensor.

6.3 Automatic extraction of features for time series

`tsfresh` [3] is a Python package for automatically extracting a multitude of features from time series data. The extracted features are filtered by testing their statistical power (for more details see XY).

We applied `tsfresh` on both the n2-norm of all accerelometer dimensions as well as a combination of the individual dimensions. Unfortunately, this didn't augment the resulting model's ability to distinguish between tram and metro trips. Even if it failed in separating tram and metro, it did help to show some new, implicit clusters though - see Figure 12. These (sub-)clusters are grouping e. g. trips in a particular transport mode by token. This wasn't achieved by DTW or any other distance measure applied directly to the time series data.

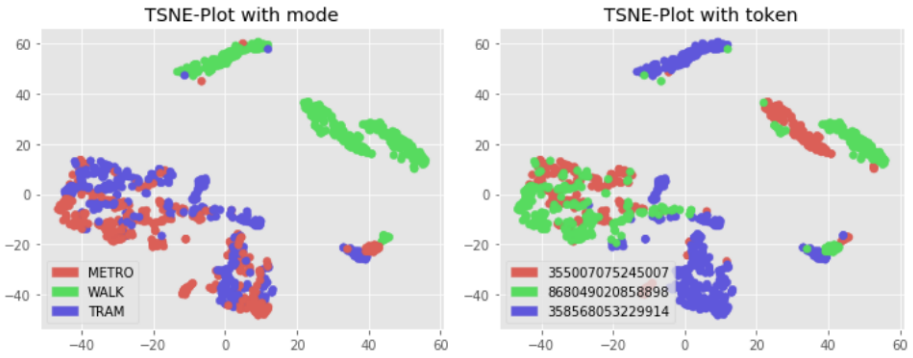

Fig. 12. Data shown in these plots is based on features extracted with `tsfresh`. *Left*: Trips reduced to two dimensions with t-SNE. Colors indicate true transport modes. *Right*: Same data, but colored according to token collecting those trips. Note that the three clusters on the top right (all of them WALK trip snippets) are actual true clusters, since they represent WALK trips gathered by the three devices in use. This pattern didn't emerge when calculating distance directly based on the time series data.

Note that a comprehensive search for features with tsfresh took a considerable amount of time - around 25 minutes on the tested machine.

7 Clustering

7.1 Calculation time

The explored clustering methods (PreDeCon, k-means, HDBScan) all took less than a minute on all tested machines (with pre-calculated distance matrices).

7.2 Clustering Algorithms Used

We established a baseline using **k-means** on previously calculated distance matrices. While we considered various clustering algorithms, we ultimately settled on implementing just one other: **HDBSCAN**. The reason for this is that we inferred from our data exploration that the biggest obstacle is the difficulty of separating trips in the tram from trips in the metro, therefore the choice of the clustering algorithm has a limited influence on the quality of the overall result. Overall we tested the following three clustering methods:

1. k-means
2. DBSCAN
3. HDBscan
4. PreDeCon

HDBSCAN is a hierarchical clustering method based on DBSCAN. It decides on a flattened clustering based on the stability of the cluster. For a detailed explanation see the paper by [2] or the very nice documentation of the hdbscan package online [5].

7.3 Applied to n2 case

k-means

First we applied a base line k-means for ordinary and scripted trips together without feature engineering and k-means with `k = 3`. This follows the naive assumption that the cluster structure is primarily influenced by the respective modes of transport.

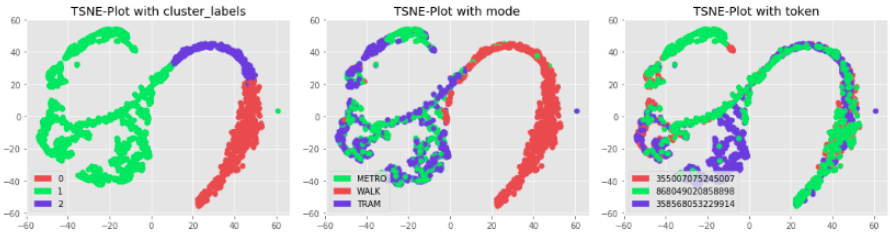

Fig. 13. TSNE plot of ordinary and scripted trips which have been cut in 30 second snippets and transformed to a euclidean distance matrix. The first plot on the left shows the clustering of our base line kmeans approach.

Fig. 13 shows the result of the clustering in a TSNE plot. We can see that the clustering treats the walking patterns as two different clusters. Since we experienced this pattern more often we investigated it. It stems from the device used by Raphael as his patterns look different than the rest of the group's.

	count_cluster_0	count_cluster_1	count_cluster_2	mode
0	0.0	659.0	2.0	TRAM
1	472.0	73.0	201.0	WALK
2	0.0	465.0	12.0	METRO

Fig. 14. Result of clustering

In fig. 14 we can see the result of the clustering in tabular form and fig. 16 shows it in a bar chart. We can see two recurring patterns. Firstly, the algorithm

[5] hdbscan documentation

could not distinguish tram and metro trips and clusters them together. And secondly, the algorithm assigns the walk patterns into 2 different clusters. Again, one of the walk patterns is solely produced by Raphael's device. To view all results of this approach please refer to N2 kmeans Notebook.

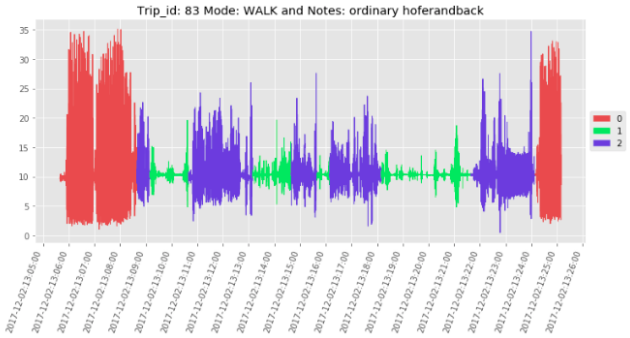

Fig. 15. Result of clustering on a single trip with our base line kmeans approach

Fig.15 shows the result of the clustering on a single trip. Start and end segment are clustered into the walk pattern cluster (the red cluster in) and the segments in between are clustered into the second walk pattern as well as the tram/metro pattern.

Results

- Estimated number of clusters: 3
- True number of clusters: 3
- Homogeneity: 0.564
- Completeness: 0.450
- V-measure: 0.500
- Adjusted MI: 0.449
- Silhouette Coefficient: 0.744

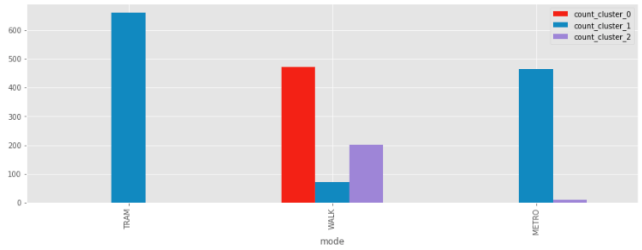

Fig. 16. Bar chart of clustering result with kmeans base line approach for n2 with euclidean distance matrix

In comparison to the base line kmeans algorithm the following two plots show the result with feature engineering and hdbscan.Be aware that for this plot the colors of the labels have switched meaning green in the TSNE plot is blue in the trip plot and the other way around. The cluster with label -1 indicates noise points identified by the algorithm.

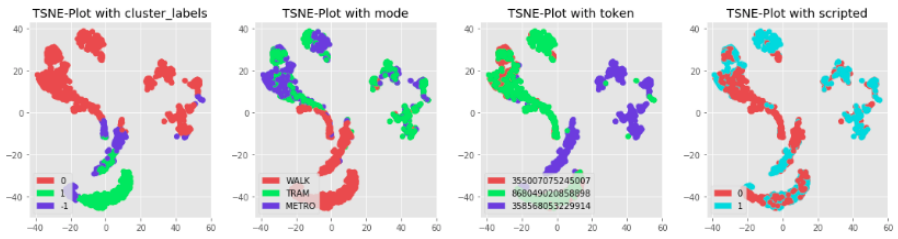

Fig. 17. TSNE plot of kmeans clustering with feature engineering and hdbscan

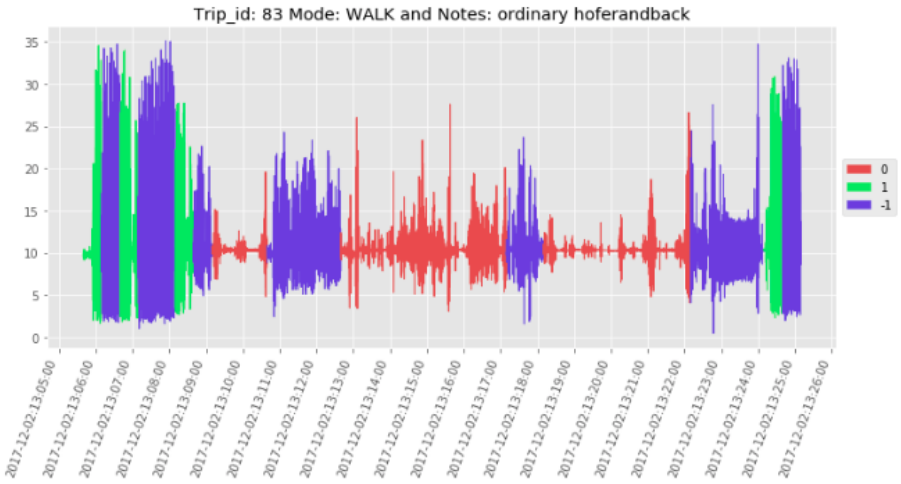

Fig. 18. Result of clustering feature engineered n2 table with all trips with HDBSCAN, here -1 indicates noise points.

Dynamic Time Warping Applying dynamic time warping to the data resulted in similar clusterings as n2, see figure 19. Again, metro and tram are indistinguishable. To see the details of this approach please refer to N2 DTW Notebook.

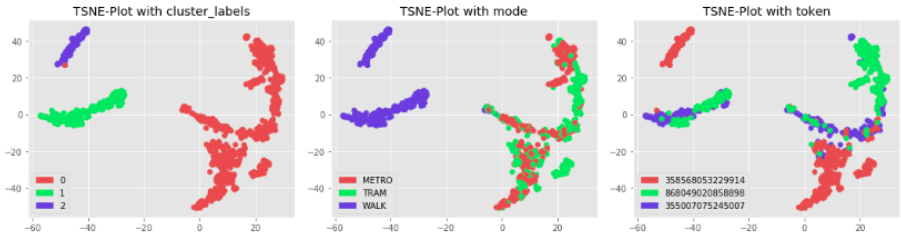

Fig. 19. TSNE of dtw clustering of n2 with kmeans

7.4 Applied to 3 dimensional case

For this model we applied the euclidean distance to the 3 dimensional data which result can be seen in fig.20. The Algorithm again cannot distinguish between tram and metro. Further details are provided in XYZ KMeans Notebook.

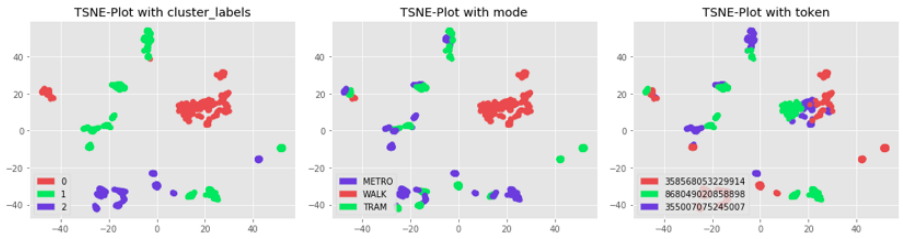

Fig. 20. TSNE of 3-dim data clustering with kmeans and euclidean distance

Below in fig. 21 the silhouette and AMI scores for various models are displayed. For example a kmeans clustering without any feature engineering of dynamically time warped n2 data resulted in a silhouette score of 0.854 and an AMI score of 0.509.

		DTW	EUCLIDEAN	
		n2		3-dim
kmeans	blank	0.854|0.509	0.744|0.449	0.601|0.439
	feat. Eng		0.441|0.296	
hdbscan			0.380|0.309	

Legend:	silhouette | AMI

Fig. 21. Overview of Silhouette and AMI scores for chosen models.The AMI scores here are calculated against the transport modes, so this measures how well the clustering captured the label structure, which of course does not have to be the real ground truth.

8 Future Work/Potential Improvements

We think that improvements should target first and foremost the feature engineering. There is a number of time series feature related to spatial data with public transport that might be utilized directly, but involve some careful preprocessing we lacked the time to include before the deadline. For example:

– A repeated or longer absence of GPS signals might indicate the metro as transport mode.

– Metros usually follow straighter lines than trams.
– Metros reach higher speed than trams, so the top speed might be a feature that could distinguish metros from trams.
– Apply filter (e. g. Kolmogorow-Smirnow, Kalman, ...) to smooth out erroneous data points before engineering features or calculating distances between time series.
– Apply techniques from signal processing to engineer features

We also think that equipping all team members with the same type of phone would be beneficial, since different devices might produce divergent data patterns (which was the case with ours). While having different devices ensure that the model is required to be more robust and less prone to overfitting, our devices were different to a point that Raphael's phone didn't even have the same type of sensors as Moritz' and Lukas', thus complicating or making impossible the engineering of some features that might have helped separating different transport modes.

References

1. Android acceleration data description, howpublished = https://developer.android.com/guide/topics/sensors/sensors_motion.html#sensors-motion-accel, note = Accessed: 2018-01-09.
2. Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 160–172. Springer, 2013.
3. Maximilian Christ, Andreas W. Kempa-Liehr, and Michael Feindt. Distributed and parallel time series feature extraction for industrial big data applications. *arXiv:1610.07717 [cs]*, October 2016. arXiv: 1610.07717.