E-Mission: Automated transportation emission calculation using smart phones

Kalyanaraman Shankari
Mogeng Yin
Shanthi Shanmugam
David E. Culler
Randy H. Katz

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-140
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-140.html

August 1, 2014

Acknowledgement

This work was supported in part by the Jim Gray Fellowship, and in part by the NSF ActionWebs CPS-0931843. We would also like to thank Akshay Vij for his invaluable support in suggesting approaches for automated inference and his insight into prompted recall. And to Thomas Raffill, for all the reviews.

E-Mission: Automated transportation emission calculation using smart phones

K. Shankari * Mogeng Yin † Shanthi Shanmugam * David Culler *
Randy Katz *

August 1, 2014

Abstract

Tracking travel patterns and modes is a goal that is useful on many levels, including calculating the transportation emissions of a population. Prior efforts to collect this information have been stymied by low accuracies or reliance on supplementary devices. In this paper, we describe a system that improves accuracy by using prompted recall on the smartphone, and aggregates the information to help detect large scale patterns. We also present the evaluation of a prototype implementation that was used to collect data from 44 users in the San Francisco Bay Area over 3 months.

1 Introduction

Transportation accounts for around 30% of US Greenhouse Gas (GHG) emissions [14]. While most efforts at reducing transporation GHG emissions have focused on automotive technologies such as electric vehicles and low carbon fuels, [5] shows that the carbon reduction goals are so aggressive that they are unlikely to be sufficient and demand reduction is essential.

For example, even if we assumed that "new fuel economy would increase to 45 mpg and fuel carbon content would decrease to 15% below current levels, then 2030 CO_2 emissions would be reduced to 1% below 2005 levels, or 24% above 1990 levels". If we need to actually *reduce* emissions to below 1990 levels, we need to reduce the amount that people drive as well.

Efforts to motivate behavior changes through outreach campaigns face both personal and structural barriers - people don't know their transportation footprint or its components, and changes to reduce their footprint are complicated in a landscape optimized for the single occupant automobile.

We propose to tackle both these barriers by tracking users' transportation modes automatically using their smart phones. This allows us to provide users with their carbon footprint, split into components, and also to understand large scale patterns and propose structural changes.

There have been several prior efforts to track user activities using smartphones. We have integrated and extended several of these efforts to build a complete end-to-end system, with apps in both the android and iPhone stores. We have used this system to collect 7439 labelled trips from 44 unpaid volunteers across 3 months.

There are two main contributions in this paper, one related to the data collection, and the other to the analysis of the collected data.

1. Our phone app prompts users to confirm the transportation mode for their trips directly on the phone. This makes recall very easy - users don't have to remember to visit a website, and can confirm trips directly from their phone, with at most two clicks. This allows us to improve the accuracy of the col-

*Electrical Engineering and Computer Science
†Institute of Transportation Studies

lected data in two ways.
   (a) Instead of building generic classifiers based on an initial phase of supervised learning, we are able to collect ground truth from each user and build personalized, user specific models.
   (b) We can prompt users only for low confidence trips. This introduces a trade-off between trip accuracy and user involvement.

   Having high accuracy data is important because [15] shows that low accuracy rates can introduce significant bias if the detected trips are used for travel demand models.
2. We aggregate individual user information to perform system level analysis (e.g. heatmaps, arrival times at work). To our knowledge, we are the first smartphone tracking project to go beyond personal tracking to aggregate, structural tracking.

The paper outline is as follows: in section 2, we compare our solution to related work, section 3 is a glossary, in section 4 and 5, we present the technical details of the prototype, in section 6, we perform an exhaustive evaluation of both the machine learning and system components, in section 7, we outline future work, and section 8 is the conclusion.

2 Related Work

There have been several efforts in the past to build automated transportation mode calculators and personal carbon footprint calculators. We provide a brief review of them here. We believe that the technology has finally matured to the point that it is feasible to use a smartphone to automatically track user activities on an ongoing basis.

Some characteristics of our system that distinguish it from earlier efforts are described below. A summary of the existing work, focusing on these features, is summarized in Table 1.

Sensors We perform the tracking using GPS data collected from smartphone sensors at relatively coarse granularity, instead of a separate GPS device with fine granularity.

Modes We automatically distinguish between motorized modes (car, bus, train, air) in addition to non-motorized modes such as walking and cycling.

Recall We allow users to correct our classifications by prompting them to confirm their trip modes directly on the phone. This has allowed us to build a large set of GPS traces labelled by user confirmed transportation mode.

Carbon We provide users with their personalized, automatically detected transportation carbon footprint, and compare it to their peers and emission reduction goals. Other carbon footprint calculators typically require users to enter their information, so are less precise, and do not provide ongoing feedback on progress towards goals.

Aggregate We aggregate individual user data in order to obtain an aggregate overview of the data. We use this to determine temporal information, such as the distribution of arrival and departure times at work, and spatial information, such as the most popular bike and car routes. To our knowledge, we are the first smartphone tracking project to go beyond personal tracking to aggregate, structural tracking.

Upload We integrate with a third party application (Moves) for the GPS data collection. This architecture allows us to potentially integrate with the existing location history tracking mechanisms that the smartphone OSes use for their personal assistants (Siri/Google Now), which allows us to take advantage of their work in optimizing smartphone battery life.

3 Glossary

1. **Trips and sections:** The data received from Moves is pre-segmented into trips, each of which consists of one or more sections. A

	UbiGreen [6]	PIER [11]	Reddy (2010) [13]	Zheng (2010) [16]	FMS [2]	PhD Thesis [8]	QT [9]
Device	separate	phone	phone	separate	phone	phone	phone
Modes	walk + cycle + transport	walk + cycle + transport	walk + cycle + transport	walk + cycle + bus + car	walk + cycle + bus + subway + motorbike + car	walk + cycle + transit + car	walk + cycle + bus + train + car
Recall	on phone	python script	offline	web	web	manual	web, optional
Carbon	green/non green	??	N	N	N	N	Y
Aggregation	N	N	N	N	??	N	N
Upload	N	auto w/ manual trigger	manual	??	auto w/ manual trigger	manual	auto
Users	14	5, 30	16,1,16	65	34, 27	6	135
Length	1-4 wks	1 day, 6 mos	15 mins, 4 wks, 1 day	10 mos	14 days	3 mos	3 weeks

Table 1: Summary of existing work focusing on features relevant to this paper

trip is a logical transition from one location to another, and may consist of multiple sections, each of which is potentially using a separate mode. Figure 1 shows a trip from work to home that consists of three sections.

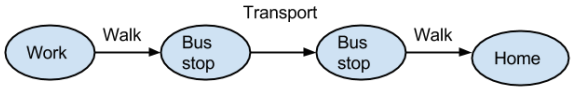

Figure 1: Examples of trips and sections

2. **Unclassified sections:** Trip sections that were detected using phone sensors but have not yet been confirmed by the user.
3. **Classified sections:** Trip sections that have been displayed to the user and confirmed as accurate or inaccurate.
4. **Predicted mode:** Mode predicted by our inference algorithm.
5. **Confirmed mode:** Mode confirmed by the user.

4 System architecture

The system architecture diagram is shown in Figure 2. The various components are briefly described below.

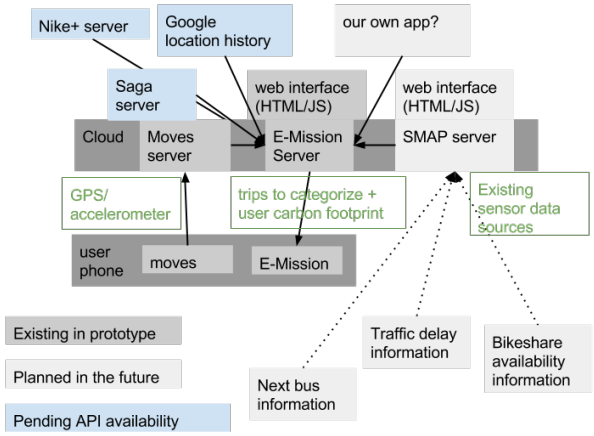

Figure 2: System architecture

4.1 Phone app

We have developed phone apps for both the android and iPhone platforms. These are available for general install using the app stores on both platforms. The apps perform 5 basic functions.

1. Authenticate the user using OAuth. We currently support authentication using google (Fig. 3), and plan to extend to other authentication providers in the future.
2. Obtain authentication to access to the data collected by the Moves app installed on the same phone. Note that we only perform au-

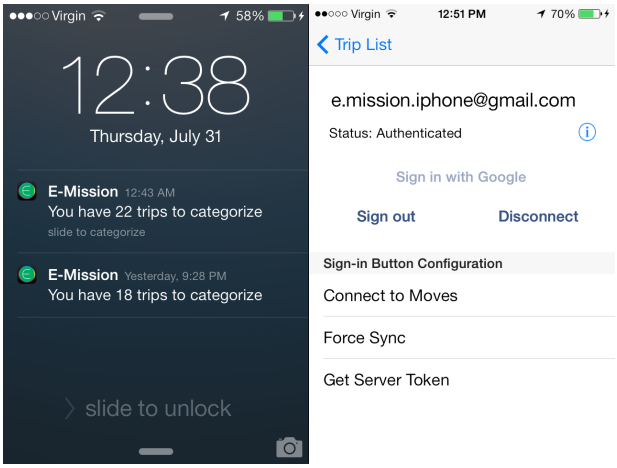

Figure 3: Example of trip notification and authentication screen

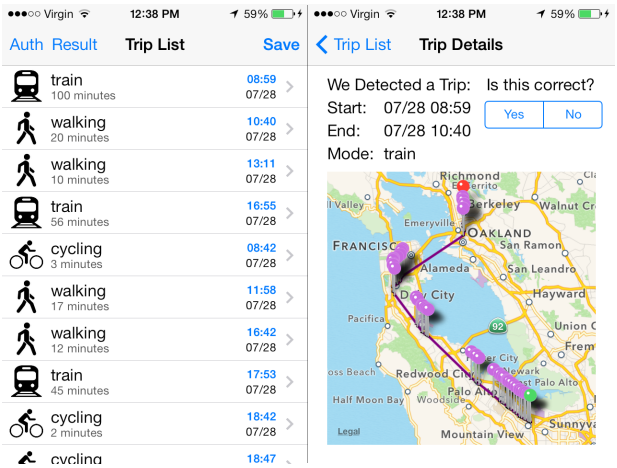

Figure 4: Sample list of trip sections and the detail of one section showing the route taken

thentication on the phone, the actual data access occurs on the server (Fig. 3).

3. Use the background sync mechanism on each individual platform to periodically read and cache unclassified trips from the server and save classified trips to the server. The mechanism used on iOS is "Background Fetch"[1], and the mechanism used on android is the "SyncAdapter"[7].
4. Display a notification prompting the user to classify unclassified trips (Fig. 3)
5. When the app is launched, display a list of unclassified trips and allow the user to confirm them.(Fig. 4)

4.2 Web app

The web app is responsible for exposing a REST API that provides access to the data in several forms. It is a fairly lightweight process that primarily reads data directly from a MongoDB instance and does not perform significant post-processing. A complete list of the current API methods is provided in Table 2. In addition, the webapp exposes a visualization UI for the aggregate functions that is built using Javascript and NVD3, invoking the REST API for the data. Selected screenshots of the web UI are shown in Figure 5.

4.3 Offline analysis scripts

In order to have a responsive interface, we perform the bulk of the processing offline in batch mode. The results of the offline processing are stored in the database for easy access by the webapp layer.

1. **GPS trace retrieval** We currently read GPS traces using the Moves app, which also conveniently breaks up the traces into trips and sections. As we integrate with other sources, we may need to incorporate trip detection algorithms here as well.
2. **Home and work location** Once we have the raw trip sections for each user, we detect home and work locations automatically. This is used for detecting commute trips.
3. **Commute mode sections** In order to support statistics on commute behaviour such as the arrival time at work, we classify trip sections as commute and non-commute.
4. **Mode inference** We use several features generated from the GPS data in order to automatically infer the mode of unclassified trips. This includes not just non-motorized modes such as walk and bike, but also, uniquely, motorized modes such as car, bus, train and air.

API name	PII?	Method	Description
`/result/commute.modeshare.distance`	N	GET	Distance travelled by each mode in commute trips
`/result/internal.modeshare.distance`	N	GET	Distance travelled by each mode inside the UC Berkeley campus
`/result/commute.modeshare/zipcode/`zc	N	GET	Number of trips in each mode for a particular zip code
`/result/commute.distance.to`	N	GET	Distance travelled during commute *to* work
`/result/commute.distance.from`	N	GET	Distance travelled during commute *from* work
`/result/commute.arrivalTime`	N	GET	Time at which users arrived at work
`/result/commute.departureTime`	N	GET	Time at which users left work
`/result/heatmap/carbon`	N	GET	Carbon intensity of various zip codes
`/result/heatmap/pop.route/cal`	N	GET	Popular routes within the UC Berkeley campus
`/result/heatmap/pop.route/commute/`$selMode$	N	GET	Popular routes for a particular commute mode
`/result/carbon/all/summary`	N	GET	Aggregate transportation carbon footprint
`/tripManager/getUnclassifiedSections`	Y	POST	The list of sections that a user needs to classify
`/tripManager/setSectionClassification`	Y	POST	User confirmed ground truth
`/compare`	Y	POST	The personalized carbon footprint for a particular user
`/movesCallback`	Y	POST	Moves auth code that is exchanged for an access token

Table 2: List of current API methods

4.4 Security and Authentication

Since our data is privacy sensitive, we have classified the methods that expose it into two groups - ones that expose Personally Identifiable Information (PII) and ones that don't. As we can see from Table 2, all methods that expose PII are HTTP POST methods, and require a JSON Web Token (JWT) for authentication. These are currently accessed from the phone apps, where we generate the JWT by authenticating with Google.

We perform two levels of authentication. We use OAuth to authenticate the user account. This allows the same user to access their data from multiple devices. We also use OAuth to authenticate with our GPS trace provider (Moves) - this gives us the permission to read the list of trips and sections that they have collected.

5 Algorithm details

5.1 Home detection

For this heuristic, we make the assumption that the first trip section made after 5am each day, has a high probability of originating from home. We combine these trips to determine the user's home location using the following steps:

1. We generate a cluster of the starting points of each trip using a distance threshold of 200 meters to account for the noise from the GPS data.
2. Then, we calculate the kernel density within the threshold for every candidate home location.
3. The location with the maximal density is defined to be the users home.

5.2 Workplace detection

Traditional techniques for detecting work location assume that the user has a relatively regular schedule at least for each weekday. Given our student-heavy sample population, we knew that users may have different working places, classrooms, etc. In order to address this, we detect the work location for each user at each weekday. We define the place that a user spends most of the time in a day (except home) as his/her work location.

5.3 Commute trip detection

Once home and work location is complete, we attempt to compute the "to" commute, which is the trip from home to work, and the "from" commute, which is the trip from work to home. The heuristic that we use for this is described below.

1. For the "to" commute, we find the first trip segment that a user made after 5am from

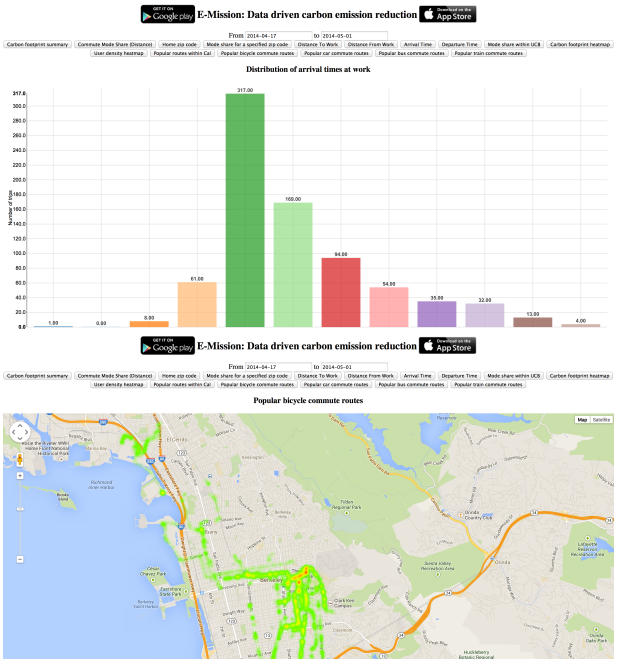

Figure 5: Arrival times at work and popular bicycle commute routes during the last two weeks of Apr 2014 (2014-04-17 to 2014-05-01)

home. This is the segment that corresponds to the user leaving home. We then iterate over subsequent trip segments until we find one that ends at work. We label this set of trip segments as the "to" commute trip.

2. Similarly, for the "from" commute, we find the last trip that a user made after 3pm that ended at home, and iterate over previous trips until we find one that starts at the work location for that particular day.
3. If the iteration for either the "to" or "from" commutes does not terminate (ie we are unable to find a trip that started or ended at work), we assume that the user didn't make any commute trips that day.

5.4 Mode detection

Our mode detection algorithm builds on the work done in[16]. We originally attempted to use the general (G) and advanced (A) features described in that paper. However, because they used dedicated GPS devices, they were able to obtain data points at 2 sec granularity. In contrast, our readings were obtained from smartphones, which meant that they were at a much coarser granularity. This was typically 30 secs, but could be as large as several minutes. This meant that the features they used, such as speed and acceleration, might not be very accurate, and so the resulting accuracies, specially for motorized modes, were low.

We augmented their basic and advanced features with spatiotemporal features. As described in Section 6.1, this allowed us to increase the accuracy for motorized modes from 30-40% to 60-70%.

1. **Bus/Train (B):** From the training set, we automatically determine bus and train station locations by looking at the start and end points of bus and train trip sections, and using the DBSCAN [4] from the `scikit-learn` library [12] algorithm to cluster them into stations. We then add features which indicate whether or not a trip section started from a known bus or train station.
2. **Location (L):** We add the lat/lng coordinates of the start and end points of the trip sections as raw features. This allows us to match against trip sections in the training set between the same pair of locations.
3. **Time (T):** We add the hour of the trip as a feature. This allows us to avoid overfitting with the location parameter. If a user uses two different modes between the same pair of locations, then adding the time makes it more likely that we will be able to distinguish between the two types of trips.

We used the `scikit-learn` [12] library to evaluate the use of various models based on this set of features.

5.5 Carbon footprint calculation

At a high level, computing the carbon footprint of transportation involves two components.

1. The distance travelled

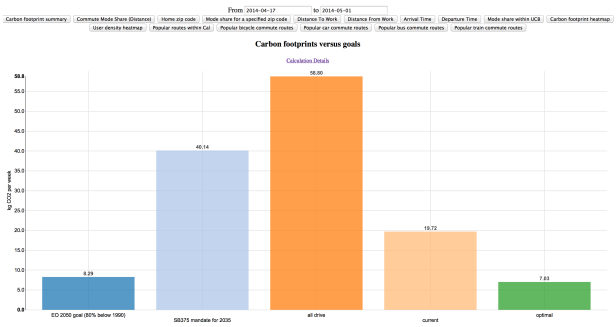

Figure 6: Aggregate transportation carbon footprint (2014-04-17 - 2014-05-01)

2. The energy consumed per unit distance

This system allows us to greatly improve the accuracy of the first piece. However, for the second piece, we just use the averages for each transportation mode from published data [10]. We use this data to calculate the following metrics, as shown in Figure 6:

1. a user's individual carbon footprint (displayed in the phone app only)
2. the average carbon footprint of the people who have installed the app
3. the footprint if the user drove everywhere
4. the California SB 375 mandate to reduce per capita carbon emissions from transportation by 15% by 2035.
5. the California EO S-3-05 mandate to reduce carbon emissions to 80% of 1990 levels by 2050. The target we show assumes a proportional reduction across sectors, and calculates the corresponding value for passenger vehicle emissions.
6. finally, we show the potential carbon footprint achievable if a user changed their commute pattern so that all short trips (< 3 miles) were through non-motorized transport and all long trips (> 3 miles) were through the most efficient motorized transport (train). This provides users a concrete and achievable goal that they can work towards.

6 Experimental results

This paper focuses the technical details of constructing a platform for estimating the transportation GHG emissions for a population. We present an evaluation of the technical details of the platform, including the learning techniques used for automated mode estimation, and the performance of the system with increasing load.

Although users consented to our privacy policy [3] by downloading the apps from the app stores, they did not provide explicit consent to having their data used for research. So this analysis will not include an analysis of their travel patterns. We expect to publish papers with that analysis in the future, which will be based on data collected with explicit consent.

6.1 Machine learning

6.1.1 Data exploration

The machine learning results are based on 7439 trip sections collected from 44 users in the San Francisco Bay Area for a period of roughly 3 months (2014-04-12 to 2014-07-18). Since the data collection was not part of a formal study and participants were not compensated for their efforts, data collection was not uniform, with participants starting and stopping collection at various times. In order to characterise the skew, the distribution of trip sections in time and across users is shown in Figure 9 and Figure 10. As we can see, although the total number of sections detected stayed relatively constant, the number of confirmed sections went down every month. This suggests that collecting background information is easier than requesting user feedback and confirmation. Further, the distribution of total and confirmed trips across users indicates that there are users who gave up on the app after a short time, users who like to run the app in the background but don't confirm trips, and users who confirm trips religiously.

The distribution of the confirmed trips is shown in Figure 11. As we can see, this is heav-

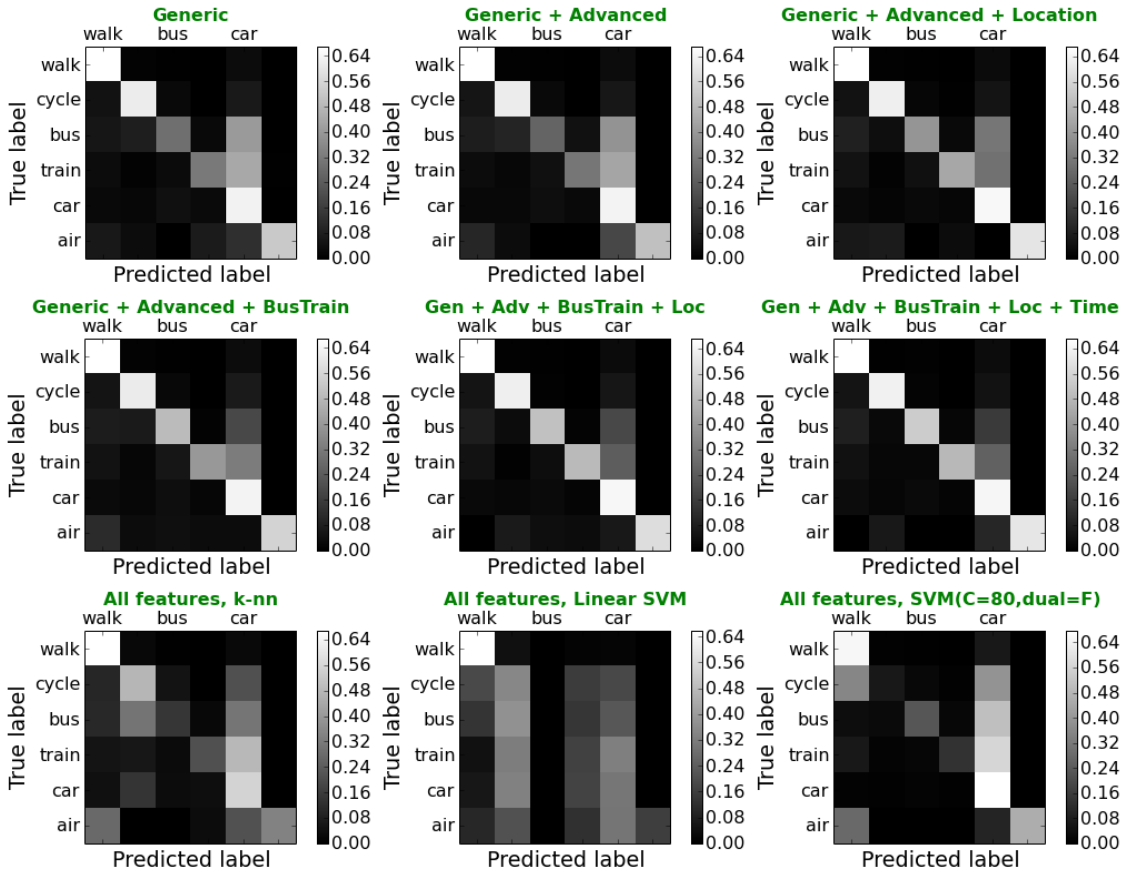

Figure 7: Confusion matrices for different combinations of features and models

ily weighted towards walking trips. This reflects both a population weighted towards students on an urban campus, and the fact that even motorized transport trips frequently involve walking at one of both ends.

6.1.2 Evaluation metrics

We do not auto-classify the `run` mode, which is rarely used for commuting, or the `mixed` mode, which is used when the Moves app did not split up trip sections correctly. Both of these also (see Fig. 11) have very few entries in our dataset. We do, however, auto-classify `air` because air trips contribute significantly to carbon emissions.

Further, since the distribution of trip modes is skewed, the overall accuracy might be a misleading metric. If the class specific accuracies are not uniform, the overall accuracy may simply reflect the proportion of high accuracy classes in the dataset. So we evaluate the accuracy of our learning methods separately for each mode. We do this by generating a confusion matrix using stratified 5-fold validation, as shown in 1.

```
for (train, test) ∈ kFolds do
    model = algo.fit(X[train], y[train]);
    yPred = model.predict(X[test]);
    cmRaw = confusion_matrix(y[test], yPred);
    // [610 12 1];
    rptSum = repeat(sum).reshape();
    // [623 623 623];
    thisCm = cmRaw / rptSum // [98 2 0];
    sumCm = sumCm + thisCm // [188 10 2];
end
avgPctCm = sumPctCm / kFolds
```

Algorithm 1: Stratified k-fold confusion matrix computation

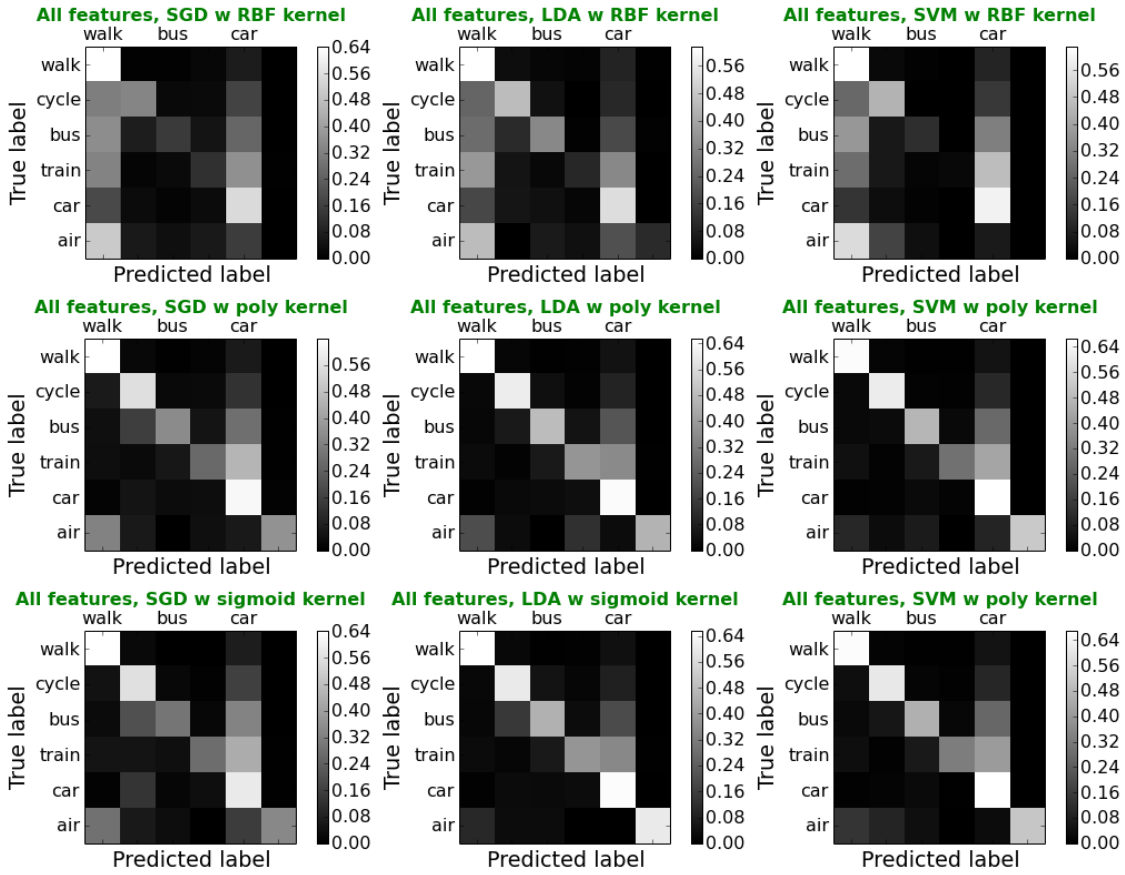

Figure 8: Confusion matrices for non-linear kernels

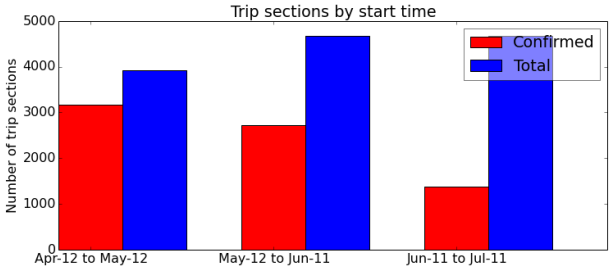

Figure 9: Number of confirmed trip sections per month

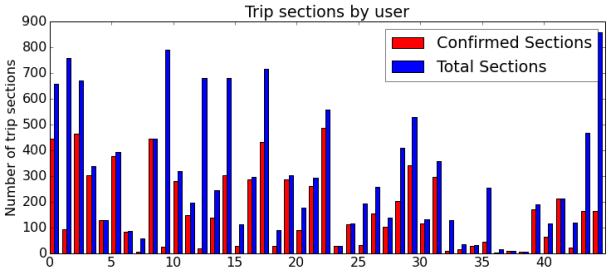

Figure 10: Number of confirmed trip sections per user

A second evaluation metric is around high confidence classifications. The phone app currently prompts the user for every trip section. In future work (Section 7, we would like to reduce the confirmation burden by prompting only for trips where the auto-classification has low probability. In order to evaluate the effectiveness of this strategy, we look at the percentage of high confidence classifications, and in order to evaluate its correctness, we look at the accuracy of only the high confidence classifications.

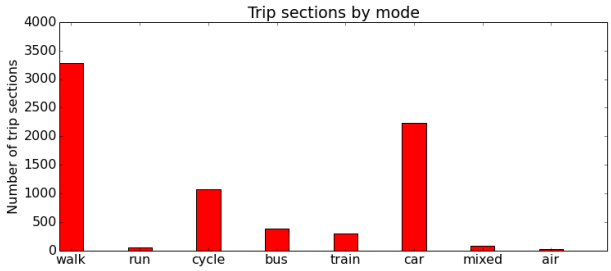

Figure 11: Number of confirmed trip sections per mode

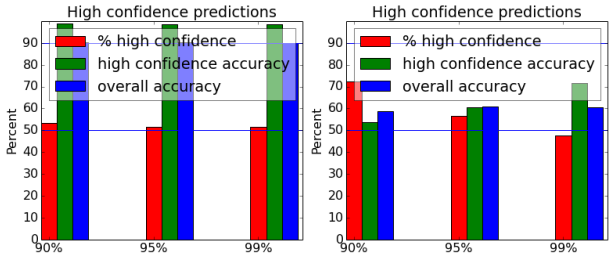

Figure 12: Autoclassification metrics using random forests(right) or linear SVM

Feature set	walk	cycle	bus	train	car	air
Generic	95	85	34	37	88	69.0
G+A	95	85	30	36	89	65.0
G+A+L	96	88	48	55	92	83.0
G+A+B	95	85	63	49	89	74.0
G+A+B+L	96	88	66	63	91	79.0
G+A+B+L+T	95	88	71	62	91	83.0

Table 3: Accuracy with different sets of features

6.1.3 Feature and model selection

There are several potential sets of features and models that we can choose from. Based on the work done in [16], we start with random forests as the model and explore various feature sets, and then we pick one feature set and validate the choice of model.

The confusion matrices for the general and advanced features from [16], and the spatio-temporal features described in Section 5.4, using a random forest as the learning technique, are shown in Figure 7.

We see here that the spatial features provide the greatest improvement in accuracy, and the best accuracy is obtained when both spatial features are combined. The current temporal feature does not improve the accuracy significantly, but does not hurt either. So, we select the **G+A+B+L+T** feature set for further analysis.

After selecting features, we evaluated the use of other learning algorithms. In [16], the other algorithms evaluated were primarily parameteric, and did not perform well. In Figure 7, we reproduced this result using a linear SVM in which the paramters were tuned using grid search. We also tried a different non-parametric method (k-nn), which was better than the parametric method, but worse than random forests.

Since the bad performance of linear models may be due to the fact that the data is not linearly separable, we also explore the use of non-linear kernels (`rbf`, `poly`, `sigmoid`) with linear models (`SGD`, `LDA`, `SVM`). A grid with these results is shown in 8. As we can see, `LDA` and `SVM` work better than `SGD`, and the `poly` and `sigmoid` kernels work better than `rbf`. However, the best results with parametric models are still worse than the random forest result, specially for the `train` mode.

Finally, Figure 12 shows us the number of high confidence trips, and their accuracy as compared to the overall accuracy. As expected, for random forests, the overall accuracy is already really high, but limiting it to high confidence trips, increased it by about 10% to almost 100%. Unfortunately, only around 50% of the trips can be auto-classified, so while we are correct, we are not that efficient.

In contrast, if we see the same metrics for the tuned linear SVM, we find that it is more efficient because there are more high confidence predictions. Unfortunately, this high confidence is not warranted - the accuracy of even the high confidence trips with a threshold of 99% is only 50%.

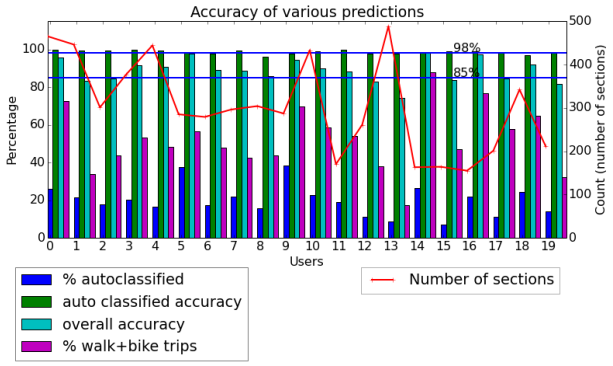

Figure 13: Various percentages for users with > 150 confirmed trips

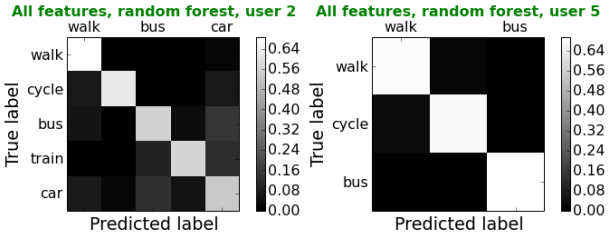

Figure 14: Confusion matrices for high accuracy user models

```
ubuntu@ip-10-217-24-94:~/Safety_Infrastructure/CFC_WebApp$ date
Mon Jul 21 06:30:53 UTC 2014
ubuntu@ip-10-217-24-94:~/Safety_Infrastructure/CFC_WebApp$ free
             total       used       free     shared    buffers     cached
Mem:        604332     590904      13428          0      42772     309932
-/+ buffers/cache:     238200     366132
Swap:            0          0          0
```

Figure 15: Snapshot of memory usage

6.2 User models

As described earlier, most prior work has assumed that an initial phase of supervised learning (training) will be used to build classifiers which will then be used for ongoing classification of large scale passive data collection.

However, as we have seen, it is hard to get high accuracies with this method. One of the reasons for this might be that there is high variability across users, which is hard to reconcile into a single generic model. For example, using a speed based feature for classification is challenging because one user is a strong bicyclist who likes to get a good workout on her commute, and the other is a beginner who likes to amble along a bike path to soak in nature. Similarly, using spatial features can be complicated if two different users typically take different modes to cover the same routes.

Given these constraints, it seemed conceivable that user specific models might provide better results. In order to test this hypothesis, we took all users who had more than 150 confirmed trips, and built user models, in which we considered only the prior trips for that user. The results are shown in Figure 13.

Note that the overall accuracy does not appear to be correlated with the total number of sections, although it does appear to be loosely correlated with the percentage of walk+bike (non motorized) trips. This is probably because, as we saw earlier, the accuracies of non-motorized modes are higher.

However, we see some cases in which the accuracy is high although the percent of non-motorized trips is fairly low. We pick two of these and plot confusion matrices for them.

The results are shown in Figure 14. As we can see, the motorized mode accuracies are higher than a combined model. This indicates that this is a promising area to explore.

6.3 System scalability

Our initial prototype system runs on an Amazon AWS micro instance with 1 vCPU and 1 GiB of RAM. In practice, since this runs on virtualized infrastructure, Figure 15 shows that the usable memory was signficantly lower.

During the construction of the system, we did not add any explicit profiling. However, we did retain all logs, and are able to reconstruct scalability curves using them. These give us an indication of the focus areas to scale to a larger system.

The metrics that we evaluated are shown in Table 4. Note that some of these metrics are

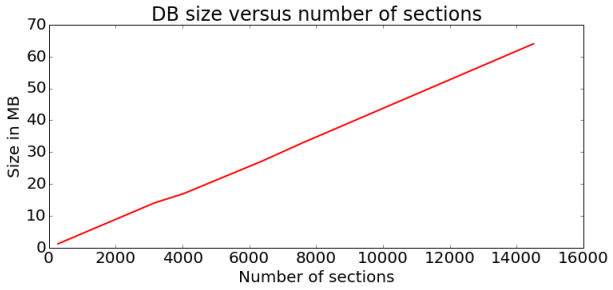

Figure 16: Size of the sections table file in MB

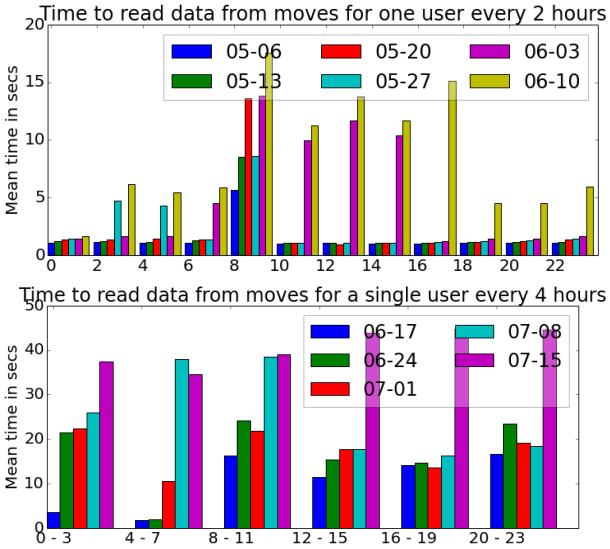

Figure 17: Mean time to pull data from Moves for a single user

generated from scripts that are run periodically using cron jobs, so we can measure the total run time in addition to the time taken for each operation. It is also easier to compare the performance across times of day for cronjobs since they run in a predictable schedule. It is harder to do this for user generated metrics, since user requests don't necessarily occur at predictable times.

Figure 16 shows the growth in the mongo DB `sections.bson` file with the growth in the number of sections. Since we store both confirmed and unconfirmed sections in the database, the database size should depend on the total number of sections. As we can see, the trend is almost perfectly linear.

Figure 17 and 18 shows the mean time for one retrieval and total run time of the data retrieval script. As we can see, this operation was initially very efficient, except for the 8am run, probably due to contention with the commute trip detection. However, around 10th June, the operation has begun to slow down at all times of the day. This is particularly bad after we turn on the machine learning pipeline. The last run (on 15th July) takes close to an hour for every run.

From Figure 19, we see that the commute trip classification scales more or less continuously until we reach a sharp discontiuity around the 10th of July. That point is not associated with any system changes, but after it, the run time for the commute sections run extends for almost the entire day. This means that the commute section script is running almost all the time.

From Figure 20, we see that the run time for the machine learning pipeline increases steadily until it takes up almost the entire 4 hours between runs. At that point, the pipeline is running almost constantly, and conflicting with the commute trip classification script above. We also see, by looking at the breakdown of one of the long runs that the step of converting the raw section data into a feature matrix dominates over everything else.

From Figure 21, we see a similar pattern with the API calls - they start off fast, go up with time, and have a discontinuity around the beginning of July. Since these are user visible response times, we plot the min, mean, max and 99% response time. The max and 99% response times are particularly concerning - the user is not going to wait for the 11 minutes that it takes to return results in the worst case, and the 30 sec max to return the section list means that 30 sec window allocated by iOS to run background fetch will be exhausted, and the app will be killed before it can display any notifications.

However, Figure 22 shows that even during the times when the response times were acceptable, the distribution of response times is neither uniform, nor does it follow a diurnal demand pat-

Metric	Invocation	Description
DB size	N/A	Size of the exported `sections.bson` file, in MB
Data retrieval	*/2, */4	Script that connects to Moves, retrives trip data for each user, and saves it to the database. Sleeps for 2 minutes after reading data for every 10 users in order to stop overwhelming Moves. Originally ran every two hours, switched to every 4 hours when the classification pipeline was enabled.
Commute sections	7	Script that reads sections for a user, determines home and work locations and commute trips, and saves the commute flag back to the database
Pipeline	*/4	Script that reads the confirmed sections as the training set and auto-classifies unclassified and unconfirmed sections
getUnclassifiedSections	N/A	API call to read the sections that need to be classified by this user
compare	N/A	API call to read the carbon footprint results for this user

Table 4: List of scalability metrics

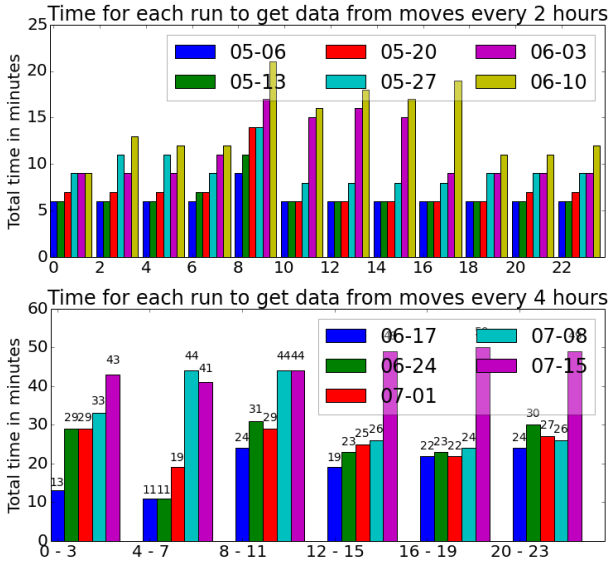

Figure 18: Run time of the script to pull data from Moves

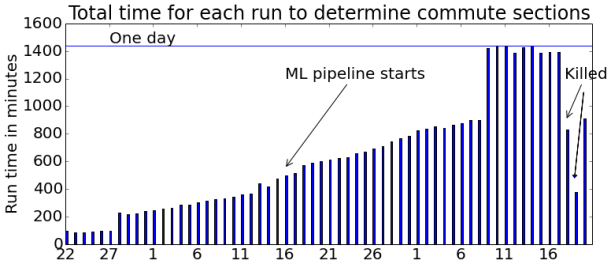

Figure 19: Run time of the script to detect commute sections

tern. Instead, the response time is higher during one contiguous chunk and negligible at other times. The higher response times probably occur during times of contention with the commute trip detection.

7 Future work

Our primary focus for future work will be on improving the phone layer, the web layer, and the analytics.

7.1 Phone layer

The primary challenge at the phone layer is to motivate people to share their trips. We need to do this by both reducing the work, and increasing the rewards. We can reduce effort by improving phone app design further, and increase rewards through some form of gamification. In addition, although the Moves team is working on optimizing power consumption, the increased power drain is still noticeable. We should consider our data needs and see if it is possible to write our own data gathering that is more optimized to our workload.

7.2 Web layer

The most immediate challenge at the web layer is that of scalability. We clearly need to move up from the micro instance, but that might not be sufficient. Note that even when we had only a small number of sections and weren't running the machine learning pipeline, the contention with

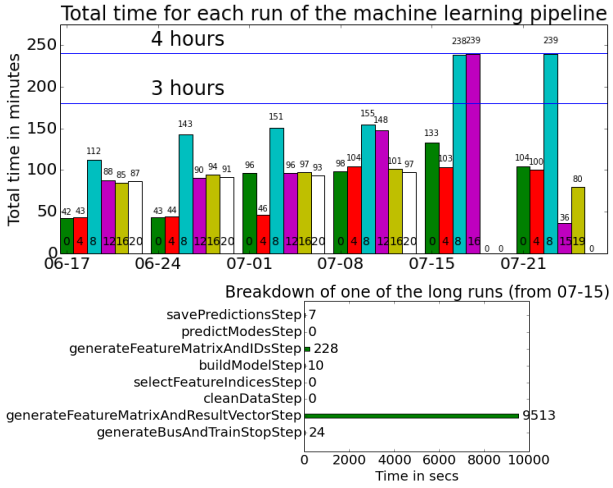

Figure 20: Run time of the script to detect commute sections

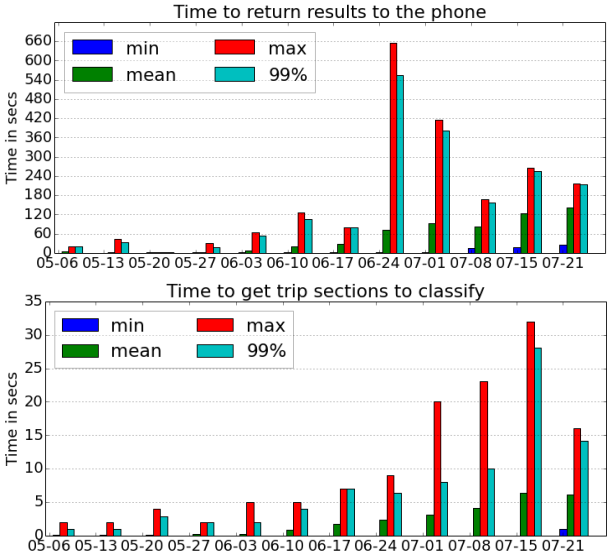

Figure 21: Response time for API calls

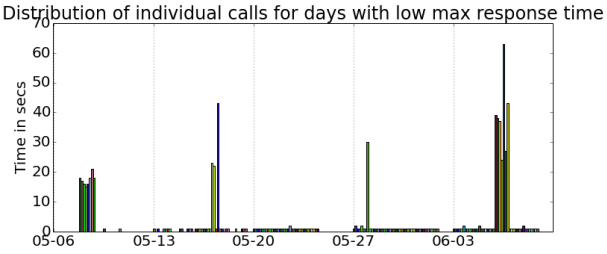

Figure 22: Distribution of the response times for the `compare` API call

the commute classification had a visible impact on both trip retrieval and API response times. We may need to either tune MongoDB, or switch to a different storage technique that will reduce the overhead caused by contention.

A longer-term challenge is that of data access and visualization. The current web app displays a subset of data that we believe will be useful at the aggregate level. However, we can easily imagine that there might be other queries that might also be interesting to other researchers. How do we change the web app to support richer visualizations, and have the option for them to be open ended? Do we support a rich query language for even more powerful access? How do we do so without sacrificing privacy?

7.3 Analytics

Finally, we want to run additional analytics to recommend actions that users and planners can take to reduce carbon emissions. We need to think of these potential recommendations, and then implement the code to detect them using external data sources. We also need to improve the carbon emission calculation to take into account more complex factors such as carpooling, fuel efficiency and so on.

8 Conclusion

We successfully built and evaluated a system to collect trip patterns for 44 users in the San Francisco Bay Area over 3 months. We are able to get accuracies of 60-95% in the automatic detecton of trip modes using a set of speed and spatial features modelled using a random forest. Prompting users for only for low confidence trips will allow us to retain high accuracies while reducing user burden. We are able to perform aggregated analysis of travel patterns in the background, with the addition of some heuristics.

Our system currently runs on an Amazon AWS micro instance, but is having issues scaling as we add more analytics. While we clearly need

to move to a bigger server, high response times caused by contention even when the dataset was smaller seem to indicate that more fundamental architectural changes may be needed.

References

[1] Apple. *iOS App Programming Guide: App States and Multitasking*. URL: https://developer.apple.com/library/ios/documentation/iphone/conceptual/iphoneosprogrammingguide/ManagingYourApplicationsFlow/ManagingYourApplicationsFlow.html#//apple_ref/doc/uid/TP40007072-CH4-SW20 (visited on 07/31/2014).

[2] Caitlin D. Cottrill, Francisco Camara Pereira, Fang Zhao, Ines Ferreira Dias, Hock Beng Lim, Moshe Ben-Akiva, P. Christopher Zegras. "The Future Mobility Survey: Experiences in developing a smartphone-based travel survey in Singapore". In: *Transportation Research Record: Journal of the Transportation Research Board* 2354 (2013), pp. 59–67. ISSN: 9780309286701. DOI: http://dx.doi.org/10.3141/2354-07. URL: http://web.mit.edu/czegras/www/TRB_FMS_Overview_Final.pdf (visited on 05/04/2014).

[3] E-Mission. *E-Mission: Data driven carbon emission reduction*. URL: https://e-mission.shankari.org/privacy (visited on 07/31/2014).

[4] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 1996, 226231. URL: http://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf (visited on 07/31/2014).

[5] Reid H. Ewing and Geoffrey Anderson. *Growing cooler: the evidence on urban development and climate change*. ULI, 2008. URL: https://www.commuterconnections.org/uploads/committee-documents/u1ZbXlk20070921140031.pdf (visited on 01/31/2014).

[6] Jon Froehlich et al. "UbiGreen: investigating a mobile tool for tracking and supporting green transportation habits". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2009, 10431052. URL: http://dl.acm.org/citation.cfm?id=1518861 (visited on 12/18/2013).

[7] Google. *Creating a Sync Adapter | Android Developers*. URL: http://developer.android.com/training/sync-adapters/creating-sync-adapter.html (visited on 07/31/2014).

[8] Jerald Jariyasunant. "Improving Traveler Information and Collecting Behavior Data with Smartphones". PhD. Berkeley: University of California, Berkeley, 2012. URL: http://escholarship.org/uc/item/69w3r763.pdf (visited on 05/07/2014).

[9] Jerald Jariyasunant, Maya Abou-Zeid, Andre Carrel, Venkatesan Ekambaram, David Gaker, Raja Sengupta. *Quantified Traveler: Travel Feedback Meets the Cloud to Change Behavior*. Tech. rep. UCTC-FR-2013-06. University of California Transportation Center, Sept. 2013. URL: http://escholarship.org/uc/item/2dh952gj#page-2.

[10] Mikhail Chester, Arpad Horvath. *Environmental Life-cycle Assessment of Passenger Transportation: A Detailed Methodology for Energy, Greenhouse Gas, and Criteria Pollutant Inventories of Automobiles, Buses, Light Rail, Heavy Rail and Air*. UCB-ITS-VWP-2008-2. Berkeley: UC Berkeley, 2008. URL: http://www.its.berkeley.edu/publications/UCB/2008/VWP/UCB-ITS-VWP-2008-2.pdf.

[11] Min Mun et al. "PEIR, the personal environmental impact report, as a platform for participatory sensing systems research". In: *Proceedings of the 7th international conference on Mobile systems, applications, and services*. 2009, 5568. URL: http://dl.acm.org/citation.cfm?id=1555823 (visited on 12/23/2013).

[12] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[13] Sasank Reddy et al. "Using mobile phones to determine transportation modes". In: *ACM Transactions on Sensor Networks* 6.2 (Feb. 2010), pp. 1–27. ISSN: 15504859. DOI: 10.1145/1689239.1689243. URL: http://portal.acm.org/citation.cfm?doid=1689239.1689243 (visited on 12/18/2013).

[14] Climate Change Division US EPA. *U.S. Greenhouse Gas Inventory Report*. en. Reports & Assessments, The national greenhouse gas inventory is developed each year to track trends in U.S. emissions and removals. Find emissions by source, economic sector and greenhouse gas. URL: http://www.epa.gov/climatechange/ghgemissions/usinventoryreport.html (visited on 07/27/2014).

[15] Akshay Vij and K. Shankari. *When is Big Data Big Enough? Implications of Using GPS-Based Surveys for Travel Demand Analysis*. Tech. rep. UCB/EECS-2014-141. EECS Department, University of California, Berkeley, 2014. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-141.html.

[16] Yu Zheng et al. "Understanding transportation modes based on GPS data for web applications". en. In: *ACM Transactions on the Web* 4.1 (Jan. 2010), pp. 1–36. ISSN: 15591131. DOI: 10.1145/1658373.1658374. URL: http://portal.acm.org/citation.cfm?doid=1658373.1658374 (visited on 05/04/2014).