Generating Natural-Language Narratives from Activity Recognition with Spurious Classification Pruning

Thomas Phan
Samsung R&D Center
San Jose, CA
thomas.phan@samsung.com

ABSTRACT

Continuous smartphone sensing offers the opportunity to perform recognition and logging of the user's activities, but a key challenge remains in conveying these activities in a manner that is both understandable by humans and useful for downstream consuming applications. We explore the viability of representing activity recognition through automatically-generated natural language English text, allowing the user to understand the activity recognition output and other software to operate on it, such as with text-to-speech software or an information retrieval search engine. To create narratives with smoother transitions and to help intelligently trigger the GPS in a power-conserving manner, we implemented a scheme called Spurious Sequential Classification Pruning that we show reduces real-time misclassifications by 76% and GPS requests by 38%. We describe a complete system for activity recognition and narrative generation and discuss its end-to-end operation.

1. INTRODUCTION

Although the ubiquity of smartphones and on-device sensors provides opportunities for performing both continuous human activity recognition [2][14][13][17] and logging [3][26], a key challenge remains in conveying these activities in a manner that is both understandable by humans and useful for downstream consuming applications. Previous efforts in activity recognition and mobile sensing include a variety of user-facing visualizations, such as metaphorical abstractions [8], location traces [5], and heatmaps [25], and while these approaches are compelling, their output is not generalizable for use by other software components.

In this paper we explore the viability of representing activity recognition results in a well-known manner, namely human natural language and specifically English. While natural language can be ambiguous, when expressed clearly it is an understandable, versatile, and useful form of information conveyance that can be leveraged by modern research and engineering in the fields of computational linguistics [6] and information retrieval [18]. We implemented a system that generates such natural language narratives from activity recognition, where both phases are performed on commodity Android smartphones. Two examples of generated narratives follow:

Example 1: *On July 15th, I started the day at 10:11 AM by driving for 10 miles from my home to Samsung R&D Center.*

Example 2: *On July 14th, I took a 2 hour, 30 minute round trip from my home. I started the trip at 6:21 PM by driving for 10 miles from my home to Subway Sandwiches on First Street where I remained for 16 minutes. I soon after went 13 miles from that point to Gates Computer Science Building in Stanford University where I idled for 17 minutes. After that, I walked from there to Oval Park. I then walked to Parking Structure 1. Finally, I drove for 23 miles to my home in San Jose.*

Such natural language representations are understandable by English speakers, and furthermore, their utility can be maximized when used in scenarios where the text is consumed by downstream software, such as: voicing these narratives with text-to-speech to allow the blind and visually-impaired to hear their activity and location traces; keeping a personal lifelog diary that can further be indexed by and queried through information retrieval search engines; and applying the text as metadata for photos or videos captured over a long vacation.

Creating an end-to-end system to generate these narratives imposes the key requirements of: (1) identifying and constraining the scope of the activity recognition narratives; (2) performing the recognition in an accurate and power-efficient manner with smooth transitions between travel legs; and (3) generating the final understandable natural language text. While our group is investigating a wide variety of human activities, in this paper we constrain narratives to clearly answer the *what*, *when*, *where*, and *how* components of user travel. *What* is the set of high-level legs of the travel found from demarcating events in traces, *when* is attained from continuous timestamps, *where* is discovered from reverse-geocoding the coordinates returned from GPS, and *how* is determined via activity recognition.

To perform the activity recognition on the phone, we built a supervised machine learning model that classifies walking, running, sitting, standing, biking, and driving using only the on-board accelerometer sensor. To reduce power consumption, we judiciously trigger GPS only between specific sequential pairs of recognized activities. Furthermore, to make the classification robust, we extract orientation-

PhoneSense'12, November 6, 2012, Toronto, ON, Canada.

Figure 1: High-level interaction between the activity recognition and the narrative generation applications.

independent features, resulting in a per-window classification accuracy of over 98%. While this accuracy is high, we saw that misclassifications occurred during real-time tests when users exhibited mixed-activity, naturalistic behavior that was not captured in our single-activity, scripted training sets. We overcome this problem and produce smoother transitions between travel legs by applying smoothing with a novel scheme called Spurious Sequential Classification Pruning (SSCP) that reduces real-time misclassifications by 76% and GPS requests by 38% at our chosen pruning threshold.

We generate the final natural language representation on the phone in three steps [23]. The activity recognition traces are first processed to determine clear demarcations between trip segments. The system then plans a coherent narrative from these segments and builds a grammatical parse data structure. Finally, the English text is realized using the SimpleNLG library [10] that applies syntax, orthography, and morphology rules to create the output.

In section 2 we describe our system and then discuss sample applications in section 3.

2. DESIGN AND IMPLEMENTATION

2.1 Overview

Figure 1 provides a high-level illustration of our system. We divide the activity recognition and narrative generation into two Android applications so that the activity recognition component can continuously generate trace data with one file per day, allowing the user to then run the narrative generator when convenient, such as at the end of the day. The final natural-language narrative can either be used directly as a written story for the user or as input to another downstream application, such as a text-to-speech engine.

2.2 Activity Recognition

The activity recognition component reads 32 Hz accelerometer samples and outputs user activity traces written to file as tuples of (epoch timestamp, activity) with occasional latitude/longitude pairs at key transitions. To determine the activities, we apply supervised machine learning to classify sliding windows of sampled data, an approach taken by previous work (e.g. [2][14]) but with novel differences which are spurred by our requirement to produce accurate, smooth transitions between travel legs.

Collecting training data: To train the machine learning model, we had users perform single-activity, scripted behavior while an Android application running on a Samsung / Google Nexus S smartphone recorded tri-axis accelerometer readings and labelled them with the user's activity, a data collection approach similar to previous work. In each resulting file, we trimmed off the first 10 and the last 10 seconds' worth of samples. There were 14 total users, all from our research lab and in the age range of late-20s to 50s. The data totalled 302 minutes, or over 5 hours, across the activities of walking (on a flat surface), walking up stairs, walking down stairs, running, sitting, standing, bicycling, and driving. We asked the users to hold the phones in any way they wanted, but most ended up placing the phone in their front right pants pocket. For sitting, users placed the phone either in that pocket or on their desk. For driving, users placed the phone either in that pocket or in a bag. For bicycling, users placed the phone in a belt-mounted hip holster.

Note that the collected data from scripted behavior resulted in a model that is able to accurately classify single activities but has difficulty during real-time tests when users perform arbitrary, naturalistic behavior that may involve either transitions between activities or completely new activities. Although we address this problem later, we note that the problem stems from the data collection stage, where gathering labelled mixed-activity traces is time-consuming. Some approaches involve reviewing videotaped users and carefully labelling activities [12], having a researcher observe users performing multiple activities and label them as they happen [14], and having users report their own activities [24]. In our work, we rely only on single-activity labelled data and address misclassifications downstream.

Feature extraction: The tri-axis accelerometer readings that we collect are orientation-dependent, so we convert them into three orientation-independent time series. For each reading we calculate the Cartesian magnitude and also project the reading onto the global horizontal plane and the global vertical axis [20]. From these three time series, we slide across windows of 128 samples with 50% overlap and extract features. The result is that the classification is performed every 2 seconds using 4 seconds' worth of data.

We compute time- and frequency-domain features, and for the latter we run an FFT and compute a correctly-scaled magnitude frequency response. Our features are: (1) time-domain power; (2) time-domain entropy; (3) the frequency f_{max} with the highest magnitude; (4) the magnitude of f_{max}; (5) the frequency $f_{weighted-mean}$ calculated as the weighted mean of the top-5 highest-magnitude frequencies weighted by magnitude; and (6) the weighted variance of the top-5 highest-magnitude frequencies weighted by magnitude. These 6 features for the 3 time series result in 18 total features per window.

Building the model: We used the Weka [11] machine learning software to build our model offline on a desktop computer. After testing several machine learning algorithms, we found that the C4.5 multi-class decision tree produced the best results, corresponding to the same choice found in other work. We applied 10-fold cross-validation, resulting in a classification accuracy of over 98% and the confusion matrix shown in Table 1. Note that while the training labels break out walking up and down stairs separately from walking, we aggregate them together into our final classifier. Similarly, standing and sitting are aggregated into an idling class. The resulting trained model is saved to a JSON file, which is then loaded into the classifier on the smartphone.

Power consumption and intelligent GPS triggering: We evaluated the power consumption of our activity recognition system on our test phones, the Samsung Nexus S and the Samsung Galaxy S II, using a Monsoon Solutions

		Predicted							
		Walking	Sitting	Running	Standing	Driving	StairsDown	StairsUp	Biking
Actual	Walking	**99.05%**	0.00%	0.09%	0.06%	0.12%	0.15%	0.34%	0.18%
	Sitting	0.00%	**99.24%**	0.00%	0.51%	0.13%	0.00%	0.00%	0.13%
	Running	0.42%	0.00%	**99.48%**	0.00%	0.00%	0.00%	0.00%	0.10%
	Standing	0.00%	1.33%	0.00%	**97.52%**	1.14%	0.00%	0.00%	0.00%
	Driving	0.16%	0.06%	0.00%	0.13%	**99.00%**	0.00%	0.00%	0.65%
	StairsDown	16.83%	0.00%	0.00%	0.00%	0.00%	**80.20%**	2.97%	0.00%
	StairsUp	29.63%	0.00%	0.00%	0.00%	0.00%	7.41%	**62.96%**	0.00%
	Biking	0.56%	0.00%	0.00%	0.00%	2.33%	0.00%	0.00%	**97.11%**

Table 1: Confusion matrix for activity recognition accuracy. Overall accuracy is 98.4%.

Power Monitor, a hardware power meter that directly provides electricity and measures the power draw. We found that the continuous recognition consistently consumes up to approximately 225 mW (or 61 mA with a 3.7 V battery) while in airplane mode with no Wi-Fi and the screen off. Further, the activity recognition itself consumes approximately 40 mW with the rest coming from the baseline CPU, which takes approximately 185 mW while idling.

As part of building a good narration, the application attempts to get the user's geolocation at key points in the travel and log it into the activity trace so that the narration generator can later resolve the location to a human-understandable description. To get the best fix for accurate story-telling, we use GPS rather than network signal trilateration. However, GPS consumes considerable power [16] [21], and so others have looked at ways to improve continuous GPS tracking. For example, [27] uses the accelerometer to sense if the phone is stationary or moving, and if the latter is found, then GPS is enabled. In [17] a Markov decision process is used to determine a continuous-GPS schedule.

In this work we take a different approach. Since the narrative generator builds travel legs between endpoints, we access GPS only at major activity transitions. To that end, we wrote a finite state machine to request GPS fixes at the transitions: from long idling (idling longer than 10 minutes) to non-idling; from driving/biking to on-foot travel and vice versa; and from running to walking and vice versa. Each GPS call has a 30 second timeout, and if a fix cannot be acquired, the location is missed and covered up by the narrative generator. Although our system can accurately recognize the above activities, the transitions are often not smooth due to spurious classifications, as we explain next.

Spurious classification pruning: A key challenge for our system is that transitions between activities must be smooth or else the narration will contain too many detailed legs of travel. One problem that arises is model overfitting. Another, as mentioned earlier, is that mixed-activity, naturalistic human behavior contains many motions that were not captured in the single-activity, scripted training data that we collected. As a result, the classifier can return an incorrect label. For example, consider the following representative example of classification results from the act of a user walking up to a car, getting keys out of his pocket containing his smartphone, and then driving off. Let `W`, `I`, `R`, and `D` be the output classification labels for walking, idling, running, and driving, respectively.

`WWWWWWDWWIIRDDDDDD`

The underlined labels correspond to the spurious classifications that occurred from the unknown motion of getting keys out of the pocket. We would like a narration of the sequence walking → driving, not walking → driving → walking → idling → running → driving. A common approach to avoiding choppiness is applying a smoothing algorithm. One type of smoothing is the use of a Markov model [22][19], but it requires ground truth transition probabilities upon which to evaluate, and as stated earlier, our assumption is that acquiring such data sets is prohibitively laborious. A simpler smoothing, which we do indeed implement, is a majority vote within a trailing window (of size 30). However, our belief is that it is better to remove misclassifications as early as possible in order to reduce pollution of the smoothing window. A resulting benefit that we will show is that by reducing misclassifications, the smoothing will also reduce extraneous GPS requests since they are made at transitions.

Leaf label	# of total occurrences	# of spurious occurrences	% valid occurrences
Driving-5	17	15	11.76%
Driving-17	1343	27	97.99%
Driving-25	25	15	40.00%
Driving-34	183	113	38.25%
Driving-37	27	13	51.85%
Driving-39	2633	99	96.24%
Driving-40	28	12	57.14%
Driving-42	183	22	87.98%

Table 2: Classification label statistics showing spurious occurrence counts of "Driving" leaves in a 9-hour trace with the percentage of valid (non-spurious) occurrences.

Looking carefully at the sequence, a key difficulty is determining why those underlined classifications may be considered spurious while other occurrences of the same classification label are not. We define a classification C to be spurious if in a window L_{pre} before C **and** in a window L_{post} after C, there are labels that differ from C. For a width of 5 for both L_{pre} and L_{post}, the underlined labels in this example sequence match this definition but the non-underlined labels do not. In our actual implementation we empirically chose a width of 10, corresponding to 20 seconds of real time; in the future we will investigate an ideal size.

Given that we can identify such spurious classifications, we apply a scheme which we call Spurious Sequential Classification Pruning (SSCP) to eliminate the problem from the model itself with judicious pruning of the C4.5 tree *after it has been built*. Looking at the tree leaves provides the intuition that we can uniquely amend each leaf label with a simple monotonically-increasing counter suffix. There are 46 leaves in the decision tree, and among them are, for example, 15 leaves with the "Driving" label. After applying this suffix scheme, these leaves then take the labels "Driving-5", "Driving-17", etc. In a subsequent evaluation phase with hold-out data, the classifier can then output these uniquely-identified labels, allowing us to determine which leaves tend to produce spurious occurrences. We can then prune them from the tree completely before shipping the software, allowing the smoother to avoid adding these spurious misclassifications into its window in real-time, real-world usage.

Table 2 shows results for some of the 15 unique "Driving" labels from classifying naturalistic, mixed-activity traces span-

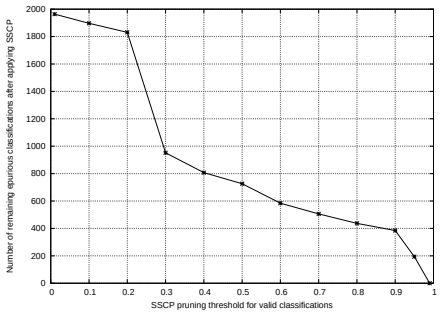

Figure 2: Resulting spurious classifications in a 9-hour trace after applying SSCP at various thresholds.

ning 539 minutes, or nearly 9 hours, from a hold-out data set. For example, the classifier emitted Driving-17 1343 times during this trace, and of those, 27 were spurious according to the definition we provided earlier, resulting in valid (that is, non-spurious) occurrences 97.99% of the time. Note that this table is ostensibly dominated by valid occurrences of Driving-17 and Driving-39, both of which occur almost always validly. The reason is that they tend to be part of long runs of true driving segments, and so their validity percentages are high. However, note that the number of such long runs is usually far less that the number of times that unknown transition periods occur. For example, in the trace for Example 2, there are only three long Driving runs (from home to Subway Sandwiches, from Subway Sandwiches to Stanford, and from Stanford to home).

By looking at the validity percentages such as those in Table 2, we can set a threshold and prune away all labels whose validity is below that value. In Figure 2 we show the remaining spurious classifications from our 9-hour trace *after* applying the SSCP scheme at various thresholds. Note that we cannot set the threshold at 100% because that would eliminate all classifications since there is always some misclassification for all the labels. For example, even the best driving classification leaf, Driving-17, has 27 spurious occurrences. In our implementation, we set the threshold at 75%, and from this trace, that would have cut the misclassifications from 1964 to approximately 471, a 76% reduction.

The impact of misclassification reduction is most evident in real-time traces at problematic transitions. We found that occasional classification flapping occurred due to using the smoothing window without SSCP, where by random chance sometimes the smoothing would result in successive changes between pairs of driving, walking, and running. Since each transition results in a GPS fix request, the flapping behavior spurs unnecessary battery usage. When applying SSCP at 75%, we avoided flapping by eliminating the labels that tend to produce a high number of misclassifications. The SSCP scheme triggered the GPS to acquire 100% of the major transitions also captured by the non-SSCP approaches, meaning that there was 100% recall, however SSCP reduced GPS fix requests by 38%.

2.3 Narrative Generation

The narrative generation application takes as input a trace of timestamped classification results from the activity recognition application and generates narratives in English. We collected over 8 hours of naturalistic travel that we set aside as a test set and showed some output in Examples 1 and 2. Some additional generated narratives are:

Example 3: *On July 15th, I took a 1 hour, 55 minute trip from work to my home. I started the trip at 2:14 PM by going from work to Lee's Sandwiches where I lounged around for 16 minutes. I thereafter went to Valley Fair Mall where I remained for 13 minutes. At that point, I went for 5 miles from there to Euphrat Art Museum in Cupertino where I chilled out for 8 minutes. I thereafter drove for 15 miles to CVS Pharmacy in San Jose. Finally, I went to my home.*

Example 4: *On May 15, I took a 1 hour, 24 minute trip from my apartment in Redmond to Savery Hall in Seattle. I started the day at 11:39 AM by scrambling from my apartment to my friend's home where I waited for 10 minutes. After that, I drove for 8 miles from there to a location at 41st Street and 15th Avenue in Seattle. Finally, I strolled to Savery Hall.*

Narrative generation proceeds through three steps [23].

Text-planning: In this phase we parse the trace generated by the activity recognition system and determine the travel legs of the trip. Each leg is defined by starting and ending points (each with a timestamp, latitude/longitude pair, and human-readable location) and a means of travel which was found from activity recognition. We identify the start and end via the intelligently-triggered GPS coordinates in the trace (mentioned in section 2.2). If a GPS request had actually timed out, then legs are essentially merged. For each GPS fix, we use Android's built-in reverse geocoding to resolve the latitude/longitude pair to a human-readable string and augment that result with Google Places, which is an online service to get nearby points-of-interest, and our own custom points-of-interest service that allows us to identify our home, work, friend's house, and other personalized locations. The end result is a list of travel legs.

Micro-planning: Given a list of travel legs, we build the skeleton of the narrative with data structures used by the surface realizer in the next phase. This skeleton comprises clauses and phrases with nouns, verbs, determiners, prepositions, and other parts of speech. For each activity, we select an appropriate verb from a custom-built list of synonyms, and to make the narrative more interesting, we add randomization during the selection. For example, the activity of "idling" can be manifest through verb phrases defined with infinitives like "to stay," "to chill out," or "to lounge about." This approach is taken for all of our recognized activities. To improve readability, we add transitional phrases such as "after that," "from there," and "finally." We further combine multiple activities into a single sentence for more fluidity. For example, if a non-idling leg is followed by an idling leg, we add the second leg as an adverbial phrase following the first, resulting in realizations like "I drove to work *where I remained for 16 minutes.*"

Surface realization: The final phase is the production of the English sentence which takes the grammatical data structure from the micro-planner and generates English text. We use the SimpleNLG surface realizer package [10], which is written in Java and available as a Jar file with a 700 kB lexicon file. Given a plan that is defined as clauses and phrases like "i", "to drive", "from home", and "to work", the surface realizer applies orthography, morphology, and syntax rules to produce "I drove from home to work.".

3. APPLICATIONS

To demonstrate the utility of the narration generation system, we implemented three application scenarios.

Text-to-speech: In the narrative generation application, we used the Android built-in text-to-speech API to vocalize the English narrative, a capability that allows the blind and visually-impaired to hear their activity traces in an understandable and straight-forward manner. Day-long traces can potentially be bland, but the micro-planning capability of our narration generator produces text that can hopefully be interesting to listen to.

Question-answering: The generated text can be used directly as a human-readable personal lifelog diary. Since such information can be voluminous, the text can then be indexed by an information retrieval search engine to allow users to issues queries over all their captured activities. When combined with question-answering [4], the natural-language data can be queried by natural-language questions, allowing the user to ask questions such as "Where do I usually drive to after work on Tuesdays?" To demonstrate the potential of this approach, we performed document- and passage-retrieval on a desktop PC by first indexing our texts with the Lucene search engine [1] and then for each question, we classified the query [15] with named entity recognition [9], retrieved documents from Lucene, retrieved passages from the documents [7], and re-ranked the resulting passages. Due to space constraints, we leave the discussion for another paper.

Metadata for multimedia: The generated narrations can be used as metadata to describe sets of multimedia (such as photos or videos), allowing them to be self-describing. Such metadata can be embedded in the multimedia itself (e.g. JPEG allows embedding arbitrary-length text) or in a separate file, thereby allowing users to search for multimedia or to provide a simple chronicle during movies or image slideshows. To demonstrate the utility of narration as metadata, we took pictures during our trips described in the sample texts and then uploaded them to the Flickr.com photo-sharing site. For each photo, we added the narrative as part of the descriptive caption, allowing not only searching but also human-readable story-telling.

4. CONCLUSION

In this paper we implemented an end-to-end system that performs activity recognition and generates natural language narratives in English. Such a representation can be understood by English speakers and consumed by downstream software, such as text-to-speech engines. To create smooth narrations and to reduce extraneous GPS fix requests caused by flapping transitions, we use smoothing augmented by a novel scheme called Spurious Sequential Classification Pruning that reduces real-time misclassifications by 76% and GPS requests by 38% at our chosen pruning threshold. In the future we will look to expand our activity vocabulary, formalize our work with SSCP, and create better narrations.

5. REFERENCES

[1] Apache Lucene, `lucene.apache.org/core/`

[2] L. Bao and S. Intille. "Activity Recognition from User-Annotated Acceleration Data," In *Proceedings of Pervasive*, 2004.

[3] G. Bell and J. Gemmell. *Total Recall: How the E-Memory Revolution Will Change Everything*, Dutton, 2009.

[4] E. Brill, S. Dumais, and M. Banko. "An analysis of the AskMSR question-answering system," In *Proceedings of Empirical Methods in NLP*, volume 10, 2002.

[5] A. Brush, A. Karlson, J. Scott, R. Sarin, A. Jacobs, B. Bond, O. Murillo, G. Hunt, M. Sinclair, K. Hammil, and S. Levi. "User Experiences with Activity-Based Navigation on Mobile Devices," In *MobileHCI*, 2010.

[6] D. Jurafsky and J. Martin. *Speech and Language Processing*, Prentice Hall, 2008.

[7] C. Clarke, G. Cormack, D. Kisman, and T. Lynam. "Question answering by passage selection (multitext experiments for TREC-9)," In *Proc. of TREC*, 2009.

[8] S. Consolvo, D. McDonald, T. Toscos, M. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, I. Smith, and J. Landay. "Activity Sensing in the Wild: A Field Trial of UbiFit Garden," In *Proceedings of CHI*, 2008.

[9] J. Finkel, T. Grenager, and C. Manning. "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling," In *Proceedings of the ACL*, 2005.

[10] A. Gatt and E. Reiter. "SimpleNLG: A Realisation Engine for Practical Applications," In *Proceedings of the European Workshop on NLG*, 2009.

[11] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. "The WEKA Data Mining Software: An Update," *SIGKDD Explorations*, 11(1), 2009.

[12] T. Hyunh, U. Blanke, and B. Schiele. "Scalable Recognition of Daily Activities with Wearable Sensors," In *Proceedings of LOCA*, 2007.

[13] J. Kwapisz, G. Weiss, and S. Moore. "Activity Recognition Using Cell Phone Accelerometers," In *Proceedings of SensorKDD*, 2010.

[14] J. Lester, T. Choudhury, and G. Borriello. "A Practical Approach to Recognizing Physical Activities," In *Proceedings of Pervasive*, 2006.

[15] X. Li and D. Roth. "Learning Question Classifiers: The Role of Semantic Information," *Journal of Natural Language Engineering*, 12(3), 2006.

[16] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. "Energy-Accuracy Trade-Off for Continuous Mobile Device Location," In *Proceedings of MobiSys*, 2010.

[17] H. Lu, J. Yang, Z. Liu, N. Lane, T. Choudhury, and A. Campbell. "The Jigsaw Continuous Sensing Engine for Mobile Phone Applications," In *Proc. of SenSys*, 2010.

[18] C. Manning, P. Raghavan, H. Schutze. *Introduction to Information Retrieval*, Cambridge Press, 2008.

[19] A. Mannini and A. Sabatini. "Machine Learning Methods for Classifying Human Physical Activity from On-Body Accelerometers," *Sensors*, vol. 10, 2010.

[20] D. Mizell. "Using gravity to estimate accelerometer orientation," In *Proceedings of ISWC*, 2003.

[21] A. Pathak, C. Hu, and M. Zhang. "Where is the Energy Spent Inside My App? Fine Grained Energy Accounting on Smartphones with Eprof," In *Proceedings of EuroSys*, 2012.

[22] L. Rabiner. "A tutorial on Hidden Markov Models and selected applications in speech recognition," In *Proceedings of the IEEE*, 77(2), Feb. 1989.

[23] E. Reiter and R. Dale. *Building Natural Language Generation Systems*, Cambridge Press, 2006.

[24] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. "Using Mobile Phones to Determine Transportation Modes," *ACM Transactions on Sensor Networks*, 6(2), Feb. 2010.

[25] I. Schweizer, R. Bartl, A. Schulz, F. Probst, and M. Muhlhauser. "NoiseMap - Real-time participatory noise maps," In *Proceedings of PhonseSense*, 2011

[26] A. Sellen and S. Whittaker. "Beyond Total Capture: A Constructive Critique of Lifelogging," *Communications of the ACM*, 53(5), 2010.

[27] Z. Zhuang, K.-H. Kim, and J. Singh. "Improving Energy Efficiency of Location Sensing on Smartphones," In *Proceedings of MobiSys*, 2010.