Activity Stalker: Recognizing and Recording What You Are Doing

Caitlin Bonnar, Felicia Cordeiro, Daseul Lee, Daniel Yang Li

{cbonnar, felicia0, dslee, dyli}@uw.edu

Abstract

We design and implement an Android application, "Activity Stalker", that can recognize and record what activity a person is doing in real time. In particular, the application can automatically distinguish between five activities: standing, walking, running, jumping jacks, and jumping rope. We use various machine-learning models and show that the classification accuracy is quite reasonable, both quantitatively and qualitatively. The application can also record the number of times a user performs a specific activity, as well as the length of each activity, either from the most recent session or the total accumulative time.

1. Introduction

Nowadays, many smart phones have powerful sensors. It would be interesting if we could take advantage of the sensors in order to create some useful applications. In this paper, we develop an activity recognition application based on the acceleration sensor, also known as the accelerometer, in an Android-based phone.

There are several motivating factors for us to develop such an application. First of all, it would be possible to monitor the health condition of a person if we developed such a system. If you sit for a long time and do little exercise, the system would record this. Perhaps another application could detect when an elderly person who is living alone falls, and respond by placing a call to emergency services. Secondly, it could monitor the behavior of children. It is not uncommon for a child to have a smart phone at present, and it could be useful for parents to understand what their child is doing. Finally, the use of such an application is very convenient, in the sense that you just need to put your phone

in your pocket and your activity data can be recorded and presented comprehensively to outline workouts and daily activities.

We did not build our system from scratch. Instead, we follow a lab tutorial from Dartmouth[1]. We generalize the tutorial in several aspects. First, we extend it to look at some newer activities (jumping rope and jumping jacks). This extension turns to be much more difficult. Secondly, we use more machine learning models, including decision trees, Naïve Bayes and logistic regression, while the lab only uses decision trees. By doing this, we can see which model performs best. Lastly, we add a new mode called history mode. That is, our application allows the user to view their activity history in a comprehensive form.

The goals for our project were two-fold: first, we wanted to determine how effective features used in the Dartmouth paper were when applied to two similar activities, jumping jacks and jumping rope; secondly, we wished to determine which classification model proved to be the most accurate and fitting for the activities we wished to recognize. What we found was that the features we chose worked pretty well, even when applied to these two extra activities. When analyzed through WEKA, we determined that a logistic regression model was most accurate; however, we were unable to test which classification model worked best in practice.

2. Related Work

Bao and Intille (Bao and Intille, 2004) are among the first to develop an activity recognition system. But their system is not practical in that it needs multiple accelerometers. Some other papers, for example Parkka et al. (Parkka et al., 2006), use multiple types of sensors. In our paper, we just use the standard smart phone's accelerometer. Kwapisz et al. (Kwapisz et al. 2010) use one accelerometer as we did, but they did not carry out user studies, nor did their system record the number of times and

[1] http://www.cs.dartmouth.edu/~campbell/cs65/lab6.pdf

the elapsed time for each activity.

3. System Overview

3.1 Data Collecting

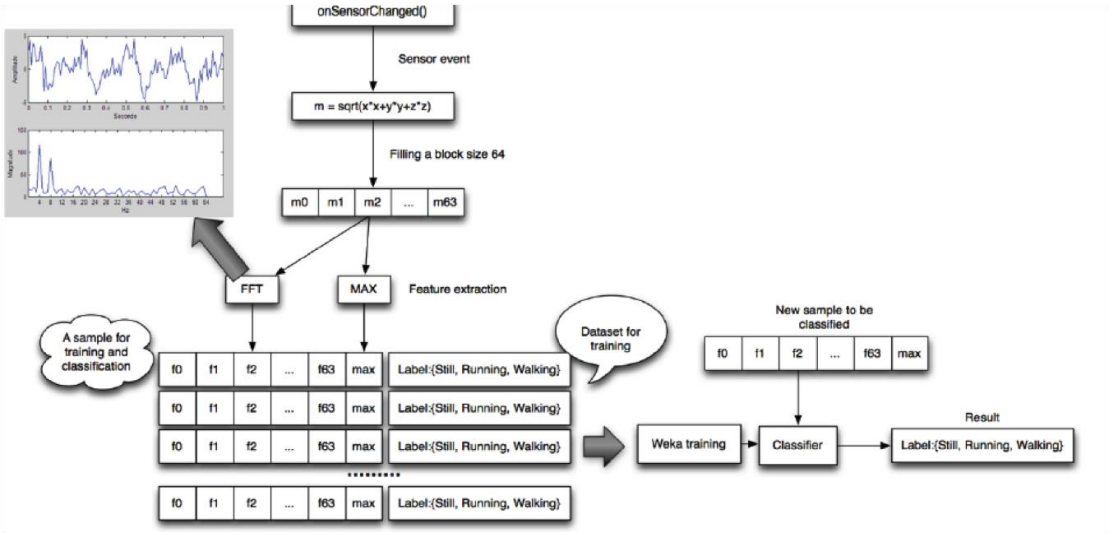

Figure 1: The data collection process[2].

The first step to our project was to collect raw accelerometer data and transform it into features that WEKA[3], the machine-learning tool that we implemented, used to train a classifier. To accomplish this, we first took in sensor samples made up of acceleration readings in the x, y, and z directions and computed their magnitudes. Once we had 64 of these, we stored them as a block and applied a Fast Fourier Transform (FFT) algorithm, which essentially converts the list of magnitudes into coefficients of a complex sinusoid and orders them from least to greatest frequency. Finally, we stored these, along with the maximum magnitude of the 64 samples and the label of the activity for which the data was currently being collected in a features file that could be read by WEKA. The reasons we chose these features were two-fold. First, the computations were efficient, and were shown to be quite accurate in the tutorial we were following. Second, the activities that we chose did not rely on direction, and so converting to magnitudes and then applying a FFT allowed for more differentiation between the accelerations produced by activities. However, something

[2] This figure is copied from [2] http://www.cs.dartmouth.edu/~campbell/cs65/lab6.pdf
[3] http://www.cs.waikato.ac.nz/ml/weka/

that we realized later was that the magnitudes between running and jumping rope proved similar enough that they were sometimes confused. Perhaps for future work we will include extra features that allow us to distinguish better between these two activities.

3.2 Training the Classifiers

The data is stored in a feature file created by the mobile device, which is then uploaded to a computer to generate a classifier. We used WEKA, one of the most popular machine learning tools for training and producing classifiers. Figures 2 and 3 are images of the WEKA application. For this project, we used only the "Experimenter" feature of WEKA. In this mode, we were able to read in the data from the features file and produce a classifier given various options and algorithms. We used the J48[4] classifier for our initial application, which is essentially a pruned decision tree that works well for recognizing repetitive tasks. It usually takes several seconds to generate the output, which contains the classifier source code, the performance (or accuracy) of the classifier, and the confusion matrix, discussed later. In addition to J48, we also used a Naïve Bayes[5] model and compared the performance and the confusion matrix against J48.

Another reason we chose J48 as our main algorithm was that WEKA was able to output source code in Java that we were then able to implement in order to automatically classify the unlabeled activities in the Android device.

[4] http://weka.sourceforge.net/doc/weka/classifiers/trees/J48.html
[5] http://en.wikipedia.org/wiki/Naive_Bayes_classifier

Figure 2: WEKA

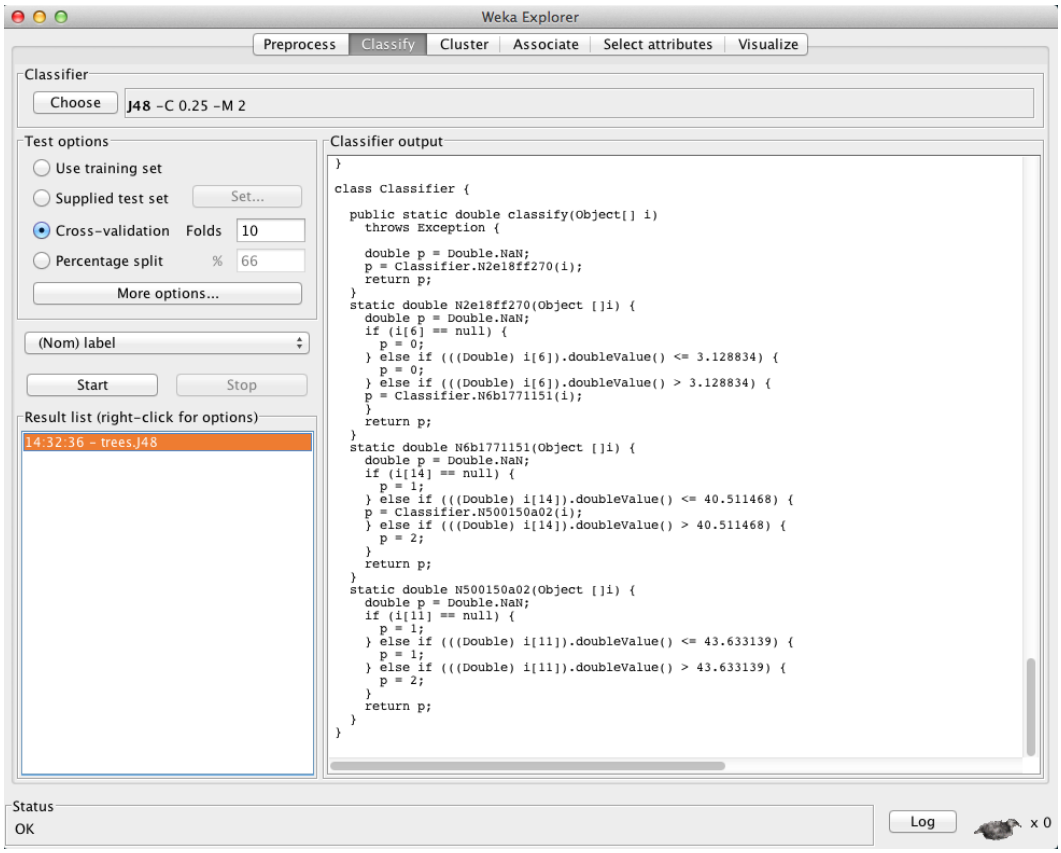

Figure 3: WEKA output for the J48 Classifier

3.3 Two Modes

For our mobile application, we used the trained classifier to create an "automatic mode" that was able to infer the user's current activity in real time, and a "history mode" which displayed most recent activities performed as well as the elapsed time

for each activity.

The automatic mode was a natural extension to the code we wrote for our data collector. The main difference was that instead of outputting data to a feature file, it is stored in a vector that is then fed into the classifier, where it interpreted the activity currently being performed and displayed it to the user. For instance, in Figure 4, when the girl is walking, our application could quickly recognize that. It takes approximately three to five seconds to read in enough sensor samples to determine activity, so there is a slight delay between readings. When there is no data initially, a "reading sensors" icon and corresponding text are displayed.

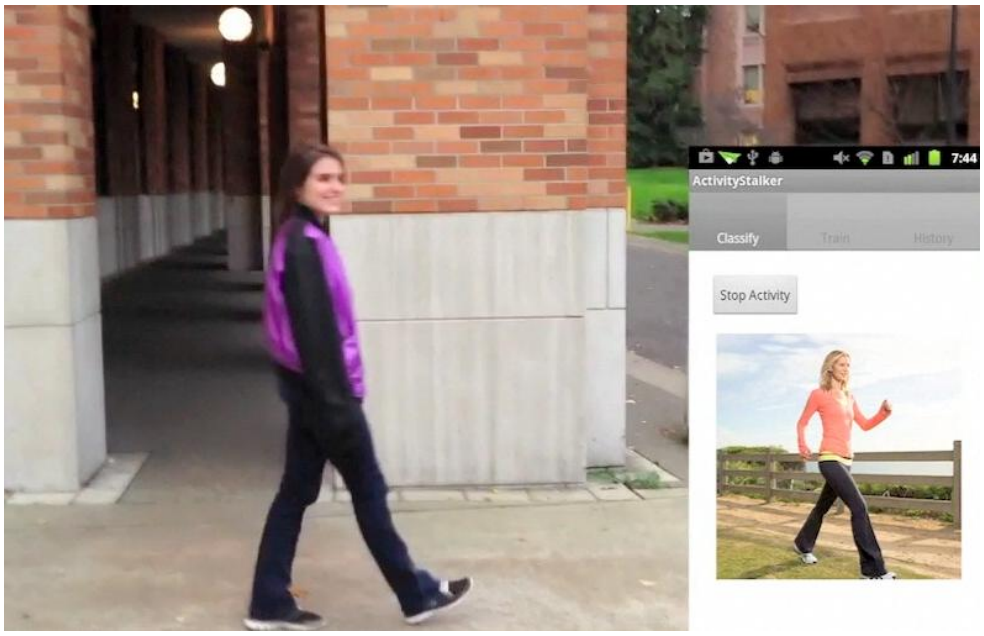

Figure 4: "Automatic mode" in action

The history mode essentially records the length of time an activity has been performed (e.g. a few minutes, hours, a day, or a week) and keeps track of the most recent activities performed. Users were given the option to reset the times recorded if they were interested in specific sessions or wanted to determine the times spent daily on each activity. A screenshot of history mode in its current state is shown in Figure 5.

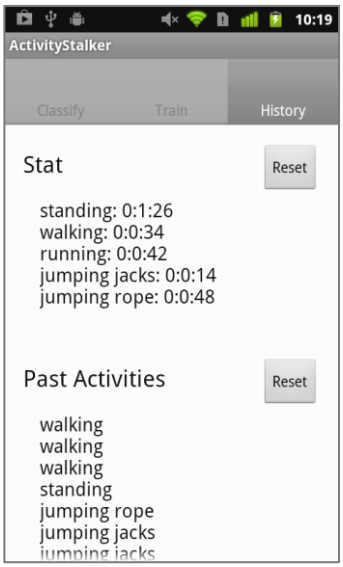

Figure 5: A screenshot of the history mode

We also wanted to incorporate various graphical displays in history mode, but we were unable to correctly implement the feature in time. Perhaps this is something that we could implement in the future, if we wanted to provide more functionality for users.

4. Experimental Studies

4.1 Classification Accuracy

We use three different machine learning models in this project: J48 (one implementation of the C4.5 decision tree), Naïve Bayes, and Logistic Regression. The results are shown as follows.

LR	S	W	R	JJ	JR
S	70	2	0	0	1
W	3	81	1	3	2
R	0	0	88	2	2
JJ	5	5	6	53	3
JR	1	3	0	0	83

Figure 6: Confusion Matrix for Logistic Regression.

J48	S	W	R	JJ	JR
S	66	5	1	0	1
W	4	74	1	2	9
R	0	0	89	0	3
JJ	0	4	2	65	1
JR	1	5	4	1	76

Figure 7: Confusion Matrix for J48.

NB	S	W	R	JJ	JR
S	70	3	0	0	0
W	17	70	0	3	0
R	0	2	83	5	2
JJ	0	2	3	63	4
JR	2	8	5	1	71

Figure 8: Confusion Matrix for Na ïve Bayes.

We performed a 10-fold cross validation on our data; that is, 90% of the data is used for training and the rest is used for testing. The matrices shown above display the instances of training data that were correctly classified based on the trained model. The rows and column headings correspond to the activity (in abbreviated form), and the diagonal entries show the number of correctly classified activities; that is, the number of instances that standing was classified correctly as standing, walking as walking, etc. All other entries denote the number of times an activity was misclassified as another activity. For instance, the Na ïve Bayes confusion matrix shows that standing was incorrectly classified as walking in seventeen instances. One possible reason for this phenomenon is that Na ïve Bayes assumes that all features are independent, and this may not be the case in practice. The overall classification accuracies are 89.372%, 86.232% and 90.5797%, for J48, Na ïve Bayes and Logistic Regression respectively.

We mainly focused on J48 in our experiment. It is simple, efficient, and effective. There is a pruning process in J48: it goes back through the tree once it's been created and attempts to remove branches that do not help by replacing them with leaf nodes. As mentioned

previously, it is also one of the few classifiers in WEKA that outputs a source code that we were able to easily copy and paste into our code.

In addition, we also compared Naïve Bayes and J48 to the Logistic Regression model, which turned out to be the model with the highest accuracy according to WEKA. We are unsure of how this model works in practice, as we were unable to implement it in our code; however, it would be interesting to compare how well these classifiers work in automatic mode for future study. Another interesting thing that we noticed was that although the confusion matrix for J48 shows that WEKA incorrectly classifies running as jumping rope three times, in practice it seemed to confuse them much more. (In fact, approximately one fifth of the time.)

4.2 User Study

Due to the time constraint, we did a very small, informal user study. The number of participants was three, and all of them were UW graduate students.

While the main reaction is that our application is interesting and easy to use in practice, the users also give us some feedback for further improvements. For example, there could be a specific style of music playing for each activity. When you are sitting, the music might be quiet and silent, and when you are running, the music should be more exciting. This may be a feature that could be interesting if the user was using the application to work out. A graphical representation of activity time would have also been a nicer UI feature.

The positive feedback for the application was that the automatic mode display (the tab labeled "Classify") was described as "easy to interpret" and was able to be seen while performing activities where the user could be holding the phone. Secondly, something we were very interested in determining was whether or not the classifier worked as well when the user was not involved in the data collection process. In other words, whether or

not the correct activity would be displayed most of the time when the current user did not collect the data that trained the classifier. Since the four members of our group did all of the data collecting in order to train the classifier, it was unclear whether our styles of running, jumping rope, or doing jumping jacks would affect accuracy. (We assume standing and walking styles are fairly similar across most users.) What we found is that the data generalized fairly well, and that the classifier was interpreting activities of users outside the group similarly to the way it interpreted them for us.

5. Conclusion and Future Work

We have developed "Activity Stalker", an Android application that recognizes the activity of a user in real time, and records the amount of time spent on each activity. In the future, we would like to do more user studies in order to further improve the application, and perhaps submit it to the Android market. Also, it would be interesting to look at direction-dependent activities and choose features that could distinguish between them. Lastly, something that we would have perhaps done differently were we to start this project again would be to establish a consistent method of collecting data; our current data is mixed between activities where the phone was being held, and activities where it remained in the user's pocket for the duration of data collection. We are unsure as to how much this affected classification accuracy, but it would be another interesting experiment to perform.

Bibliography

Bao, L. and Intille, S. 2004. Activity Recognition from User-Annotated Acceleration Data. Lecture Notes in Computer Science 3001, 1-17.

Parkka, J., Ermes, M., Korpippa, P., Peltola, J., and Korhonen, I. 2006. Activity Classification using realistic data from wearable sensors. IEEE Trasactions on Information Technology in Biomedicine, 10(1), 119-128.

Kwapisz, J., Weiss, G., and Moore S., 2010. Activity Recognition using Cell Phone

Accelerometers. SIGKDD Exploration, 12(2), 74-82.

Appendices

1. **Which people did what parts of the work? Were there problematic dynamics in your group?**

   We mostly split the tasks and design; Daseul did most of the coding for data collecting/automatic mode, Felicia worked with history mode and found the lab we followed as well as made important design decisions; Caitlin created the presentation and graphics, worked with the feature generation aspects and WEKA; and Daniel wrote the paper, performed the user study, and compared Naïve Bayes, J48, and Logistic Regression. There were no problematic dynamics other than the fact that it was sometimes difficult to coordinate meeting times between four people.

2. **Anything you considered surprising or that you learned. What would you do differently if you could?**

   What we found surprising was that jumping rope was often more confused for running than doing jumping jacks. It was also surprisingly difficult to create the type of visual graphs we wanted in history mode. If we were to do anything differently, we would play around with different features in order to determine the most effective and see how well they did at distinguishing between running and jumping rope.

   Something we learned was that seemingly very different models ended up all classifying fairly accurately. If we had more time, we would also figure out how to implement these classifiers in our application in order to do practical analysis. Learning how to generate features based on raw acceleration data was also an interesting part of this project. Determining whether or not they apply to additional, more complex activities was something that was difficult to predict. Lastly, we all learned how to use WEKA (this was the first time any of us had ever used it), which will perhaps prove useful in our future research projects.