Do we dream or do we have a nightmare?

Joshua Antonson
Carnegie Mellon University
Department of Electrical and Computer Engineering
Pittsburgh, PA, USA
jantonso@andrew.cmu.edu

Abstract

Sleep is fundamental to the overall health and well being of humans. In this paper, I present a series of methods to analyze the sleep/wake pattern of a user through a simple android application. By recording accelerometer data during a night of sleep, I can identify user movement and periods of restlessness, when the user wakes up, and label the different sleep stages. Preliminary results confirm the ability to identify user movement and periods of restlessness, as well as when the user wakes up. In future iterations, I will add the ability to label the different stages of sleep (light, deep, REM).

Introduction

Sleep is fundamental to the overall health and well-being of humans and is therefore a widely studied phenomenon. There are five stages of sleep: 1, 2, 3, 4, and REM (rapid eye movement) sleep. We cycle through the stages from 1 to REM, and then repeat. Stages 1 and 2 are classified as light sleep, stages 3 and 4 are referred to as deep sleep, and stage 5 is REM sleep. As mentioned above, when we fall asleep and how we sleep is largely affected by our behavior leading up to falling asleep. The more consistent our routine, the better we sleep. In addition, environmental factors, such as temperature, darkness, and quietness affect how we sleep. Being able to identify the sleep/wake behavior allows us to determine the restlessness of a user and their different sleep cycles. From this, the user can see their sleep habits and how well they are actually sleeping.

In order to analyze the sleep cycles of the user and their overall sleep, we must be able to identify the general movement of a user during sleep. A well-known method of doing so is to use accelerometer data collected by sensors during the course of sleep. My approach is to use an android application placed on the bed to record accelerometer data, which allows me to collect accurate data without imposing any new technology onto the user such as wearable sensors or sensors in the bed. From this, I can distinguish between the user lying still on the bed versus when they make a noticeable movement such as rolling over in bed. I then analyze the periodicity of the user's sleep movements and train a classifier in order to recognize the different stages of sleep. This is very challenging using only movement related data, as a large piece of the different sleep stages is different bodily behavior such as varying brain waves, body temperature, and heart rate.

My approach accurately identifies and classifies movement events during sleep, recognizes when a user wakes up, is restless, or is asleep.

Previous work

I will briefly describe a few approaches to the problem of analyzing the sleep/wake pattern. In relation to my work, these fall into two categories: applications that monitor the sleep behavior of a user and algorithms/studies that analyze movement events and the different phases of sleep using accelerometer or other sensor data.

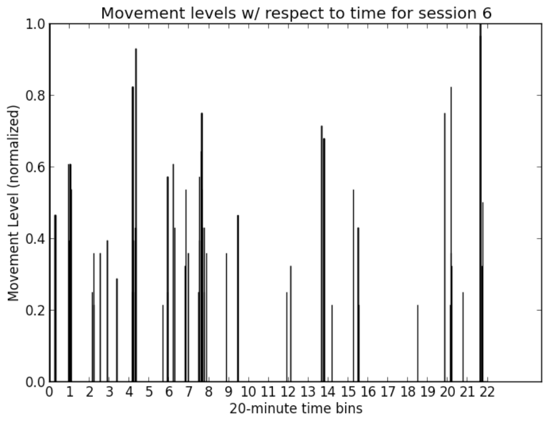
Most of the current sleep apps focus on waking the user up at the optimal time. Sleep Bot and Sleep Cycle are two popular sleep tracking phone applications. They use accelerometer and other movement related data to estimate the sleep phases of the user. The user can set a target time to wake up, and then the app will wake the user when they are in the light sleep phase and near the target time. They can also see a time based graph which displays the different sleep phases. This is closely related to my approach, but I am also focusing on collecting and analyzing data related to user behavior during sleep.

There are numerous algorithms and studies related to using movement data collected during sleep to estimate the sleep/wake pattern. I will briefly discuss one that relates closely to my approach. During this study, three professors at the University of Virginia collected movement data using the WISP platform: active RFID-based sensors equipped with accelerometers. They compared the results using these WISP sensors to both pressure sensors embedded in a mattress and to a simple iPhone application, which is similar to my method. Their method was 100% accurate in detecting discrete movement events during sleep and is 90% accurate at determining body position on a bed. This study is one of many in the field of using sensor data to analyze the sleep/wake patterns.

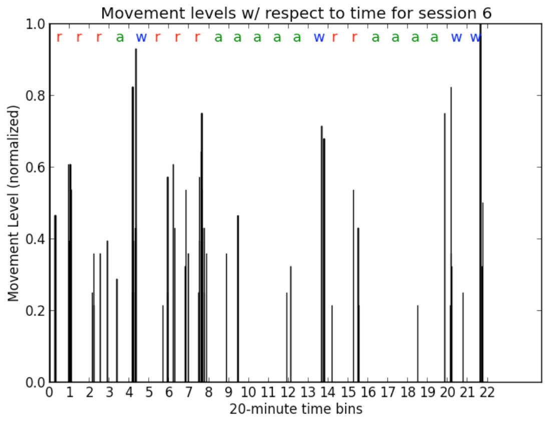
Approach

I built an Android application that collects accelerometer data during sleep. The user leaves the phone on their bed while they sleep and it records the accelerometer data during sleep. The accelerometer sensor records five measurements per second and records the x, y, and z acceleration measurements in m/s^2. Every minute, the application sends the data for the prior minute time period to a server running in Google App Engine, which stores it for processing at a later time. The accelerometer in a user's phone is not the most accurate method to measure movement patterns, especially since the phone is placed on the bed next to the user. A more accurate method would be to place sensors on the user or cover the mattress with an array of sensors, but by using the accelerometer within the phone I can create an inexpensive and unobtrusive method to collect reasonably accurate movement data.

Once the user wakes up, the app will display information to the user regarding their sleep/wake pattern via a time-based graph of movement events. The following is an example of a user's time-based graph of movement events for a sample night of sleep. I will explain what the movement events correspond to shortly, but intuitively they capture the user's movements during a night of sleep.

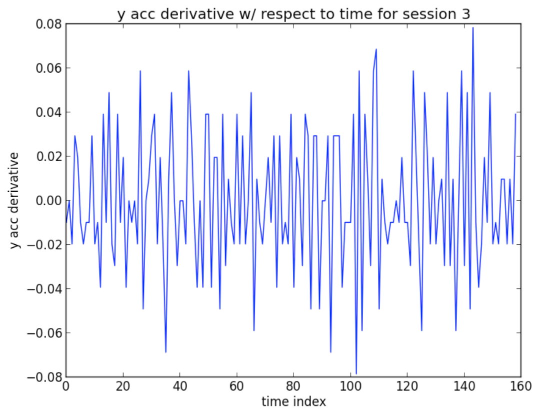


They will then label key points during their sleep. The acceleration data is broken up into 20-minute time bins and the user labels each time bin as "restless", "woke up", or "asleep". A "restless" label corresponds to a period of time when they are attempting to fall asleep, possibly tossing and turning. A "woke up" label corresponds to a period of time when they woke up and an "asleep" label corresponds to a time period when they were asleep. Clearly, we are making a very large assumption that the user accurately labels their sleep. In addition, a user knows when they woke up, but they are essentially estimating when they fell asleep in order to distinguish between a "restless" label and an "asleep label", so we must keep that in mind when accounting for the accuracy of our approach. The following is an example of the user's labeling for the previous time-based graph of movement events with "r' corresponding to "restless", "w" corresponding to "woke up", and "a" corresponding to "asleep".



My approach consists of three key steps. The first is data collection, which was just discussed, and consists of gathering acceleration data from the user while they sleep and having them label each 20 minute time window from the previous night of sleep. Second, from the acceleration data I can identify and extract movement events during a night of sleep for the user. Finally, I can classify the sleep according to these movement events and the user's labels.

In order to be able to classify the sleep/wake patterns of a user we need to be able to identify movement events. Intuitively a movement event corresponds to a movement made by the user while they sleep, i.e. rolling over in bed, sitting up, getting out of bed, etc. In order to identify and extract movement events we need to gather baseline accelerometer data that corresponds to no movement by the user. From this baseline data, we establish threshold accelerometer values for a given user's movements. I have the user place their phone on the bed and then have the user lay on their right side, left side, stomach, and back and also have the bed empty. For each of these sessions, the user stays still. From this, we gather accelerometer data related to no movement for the user in a wide range of different scenarios. The derivative of acceleration, for x, y, and z and with respect to time, stays within a certain threshold [-b1, +b1] for each of the respective scenarios. Of note, is the fact that the threshold for the x, y, and z accelerations can be different depending on the user's bed and phone. The following figure shows that the y acceleration threshold for a given user and no movement is [-0.08, 0.08] for a given scenario, i.e. phone on left middle of bed and user lying on right side.



Once I have established x, y, z threshold values for no movement in each of the different scenarios for a user, I can identify movement events over the course of a night of sleep. I consider a movement event as when one of the acceleration derivative readings exceeds the threshold values established in the previous step. So, for example if the x acceleration derivative at a time step was 0.15 and the threshold was [-0.08, 0.08] as defined previously, then I would consider that time step as a movement event. For each two-second window, I define the movement weight of the window as the number of movement events during that window. So for each timestamp that the x, y, or z acceleration derivative exceeds the threshold within that two-second window, I increase the movement weight by one. Then, I normalize the weight of each window by the maximum weight of all the windows. We then remove all of the windows where the normalized weight is less than 0.20 as these weights correspond to subtle movements that do not affect sleep. I am only concerned with identifying large movements such as rolling over, sitting up, getting out of bed, etc. From this, we are left with two-second time windows that are weighted according to the amount of movement during that time frame

Next, I cluster windows of movement in order to identify ten-second periods with multiple movements. If there are two or more two-second windows of movement within a ten second window, then I assign the five two-second windows within that ten-second window to have the weight of the maximum two-second period weight. From this, we have identified periods of clustered movement events throughout the user's sleep. The first figure from the previous page shows the movement levels (normalized) over the course of a 10-hour period of sleep for a user and is what the user sees once they are finished with a night of sleep, as mentioned previously.

The third key step is to classify the user's sleep. In this step we attempt to label each of the 20-minute time bins as "restless", "asleep", or "woke up". We train a classifier using 20-minute time bin of movement events and the resulting user generated labels from a training set of data. The training set of data comprises multiple nights of sleep for multiple users. The classification algorithm that I am using is k-NN with k = 1. The k-NN algorithm relies on a similarity measurement, so the problem boils down to defining a metric to measure the similarity between 20-minute time bins of movement events. For each new 20-minute time bin, I find its closest neighbor using the similarity metric and assign it to the label of its neighbor. Currently, I have used two different methods for determining the similarity of 20-minute time bins of movement events.

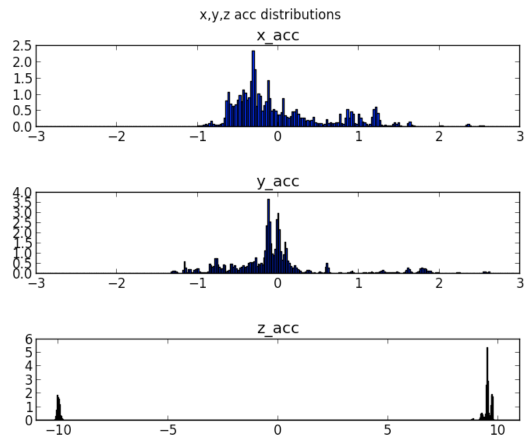
The first method is called dynamic time warping. Dynamic time warping attempts to find the optimal non-linear alignment between two time series. It essentially shifts one time-series over one step at a time and computes the Euclidean distance between the two time series for each alignment. The similarity measure for the two time series is then the minimum Euclidean distance, i.e. the optimal non-linear alignment. The drawback to this method is that the algorithm is quadratic in the length of the time series, but it can be sped up using a locality constraint. The assumption of the locality constraint is that it is very unlikely for two time steps in the respective time series, ti and tj, to be matched if i and j are very far apart, as determined by a window size w. Thus, it only needs to consider mappings that are within a given window size w. More information on dynamic time warping can be found at [5].

The second method is to "compress" the 20-minute time bins of movement events by extracting only its crucial information. I compress each 20-minute time bin of movement events, which has 600 movement levels data points, into a 4-tuple of four defining metrics. The four metrics are the mean of the movement levels, the maximum movement level, the number of non-zero movement levels, and the length of the biggest consecutive block of non-zero movement levels. Now, we can define the similarity between two 20-minute time bins of movement events as the Euclidean distance between their respective 4-tuples of defining metrics.

Finally, we use the trained classifier to label each 20-minute time bin for the future sleep of a user as "restless", "woke up", or "asleep" in order to give them a better understanding of their sleep/wake pattern.

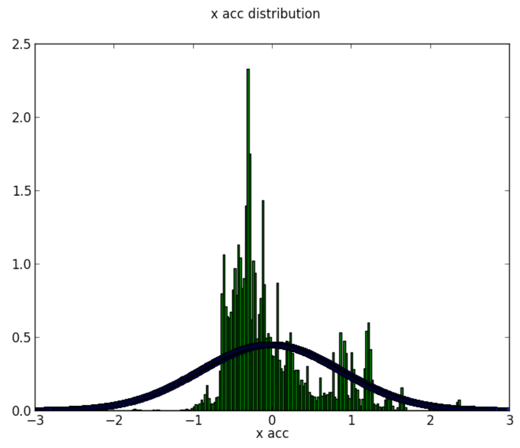
Experimental setup and results

So far, I have collected data using the application myself and five other users. For myself, I have collected 6 nights worth of data. Each of the other five users has collected data over 2 nights. The following figure shows the distribution of the x, y, and z acceleration data for one night, chosen at random, from each of the users.

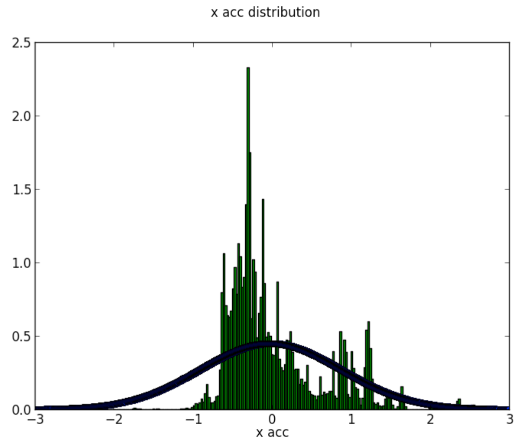


Intuitively, we can see that the x-acc distribution and y-acc distribution look to be Gaussian, and the z-acc distribution is mixed Gaussian with one centered on 9.8 and one centered around -9.8.

The following figure shows the x-acc distribution with the corresponding Gaussian fit with mean of -0.0185 and standard deviation of 0.886.

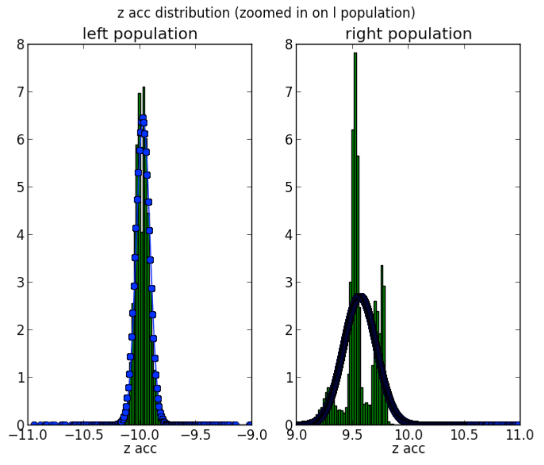


The following figure shows the y-acc distribution with the corresponding Gaussian fit with mean of -0.0188 and standard deviation of 0.706.



The Gaussian fits for the x and y accelerations have very large standard deviations with a mean right around 0.0. This is because when the user is lying still on their bed, the acceleration value is right around 0.0. When they make a movement, the value spikes up to between 1.0 and 2.0 or -1.0 and -2.0. Since the majority of the time the user is lying still, there are a lot more values close to 0.0 than above 1.0, hence the mean close to 0.0 but the large standard deviation.

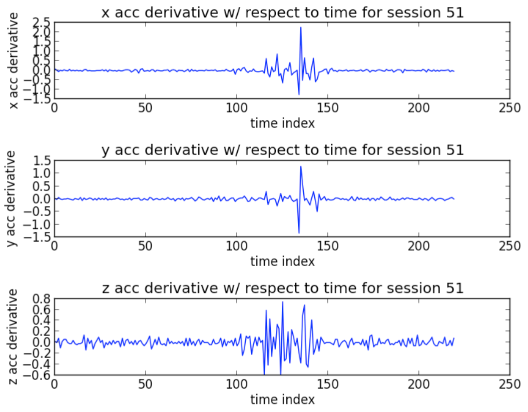
The following figure shows the two populations for the z-acc distribution with the corresponding Gaussian fits with mean of -9.97 and standard deviation of 0.06 for the left population and mean of 9.56 and standard deviation of 0.14 for the right population.



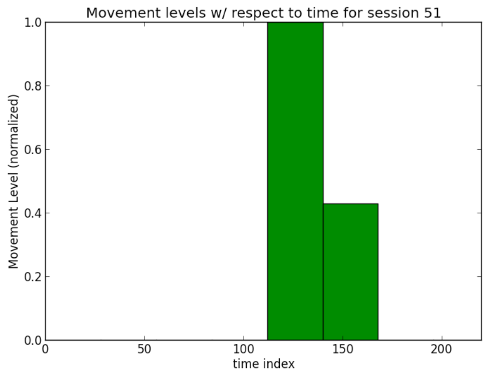
We are collecting acceleration data, so it makes sense to see the z-acc distribution be centered on 9.8, as that is the value of acceleration due to gravity. I believe that we are seeing the two populations centered on -9.8 and 9.8, respectively, because the orientation of acceleration data is unique for each phone. For the z-axis, which is perpendicular to the bed, we expect the value to be fairly consistent around 9.8 or -9.8, depending on the phone, as the likelihood of the phone flipping over mid-sleep is very low. This also explains why the standard deviation is a lot less for the z-acc distributions then the x and y distributions, as changes in z-acc are much less due to the z-axis orientation.

In addition, I have had each of the six users record five 10-second baseline sessions, where they lay still on their bed in each position: on left side, on right side, on back, on stomach, and empty bed. From this, I established the threshold ranges for x, y, and z acceleration derivatives which is used to identify movements, as discussed in the approach section.

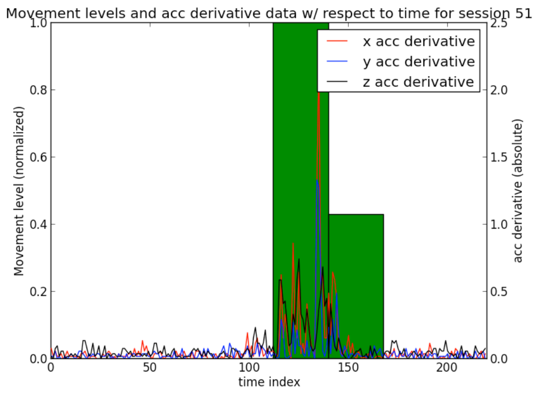
In order to test the accuracy of my approaches identification of movement events, I have had each user record four 15-second sessions, where they lay still at the start then make a noticeable movement in the middle, such as rolling over, and then lay still at the end. The following figure shows the x, y, and z acceleration derivatives with respect to time for one of these sessions, i.e. rolling over from left side to right side.



As you can see in the figure, there is one distinct spike in the x, y, and z acceleration derivative data that corresponds to the user's movement. The following figure shows my algorithm's attempt to identify the movement levels for each two-second window of this session. Each bar corresponds to the movement level (normalized) for each two-second window.



The following figure shows the x, y, and z acceleration derivative data overlaid on top of my algorithm's identification of movement levels for each two-second window of this session. Note: the acceleration derivative is absolute value in this figure.



As you can see from the figure, my algorithm successfully identifies the one movement event, as the movement levels spikes when the acceleration data spikes during the user's movement and is 0.0 when the acceleration data corresponds to no movement.

Out of the 24 of these single movement sessions, my approach correctly identified 22 movements for an overall accuracy of 91.66% and didn't report any false movements. From this, we can reasonably assume the accuracy of identifying movements during a user's sleep.

Now, I will discuss the results that I have obtained related to the classification of labeling the user's sleep. There are 300 20-minute time bins of movement events from all six users. I have split up the six users into two groups: training and testing. Training consists of myself and two other users, comprising a total of 10 nights worth of data and 261 20-minute time bins of movement events. Testing consists of the final two users, comprising a total of 4 nights worth of data and 39 20-minute time bins of movement events. The following chart shows the classification results for method 1.

```
total accuracy = 0.435897435897

asleep accuracy (a) = 0.647058823529

restless accuracy (r) = 0.0833333333333

woke up accuracy (w) = 0.5

             precision    recall  f1-score   support

          a       0.58      0.65      0.61        17
          r       0.20      0.08      0.12        12
          w       0.33      0.50      0.40        10

avg / total       0.40      0.44      0.41        39
```

Method 1, which is the k-NN classifier using dynamic time warping, achieved a total accuracy of 43.59%. Method 1 was able to recognize "asleep" labels with 64.71% accuracy, "restless" labels with 0.08% accuracy, and "woke up" labels with 50% accuracy.

The following chart shows the classification results for method 2.

```
total accuracy = 0.589743589744

asleep accuracy (a) = 0.823529411765

restless accuracy (r) = 0.416666666667

woke up accuracy (w) = 0.4

             precision    recall  f1-score   support

          a       0.74      0.82      0.78        17
          r       0.45      0.42      0.43        12
          w       0.44      0.40      0.42        10

avg / total       0.58      0.59      0.58        39
```

Method 2, which is the k-NN classifier using 4-tuple of key metrics, achieved a total accuracy of 58.97%. Method 2 was able to recognize "asleep" labels with 82.35% accuracy, "restless" labels with 41.66% accuracy, and "woke up" labels with 40% accuracy.

From this, we can clearly see that method 2 is a lot more accurate for both recognizing “asleep” and “restless” labels, with only a slight decrease in the accuracy of recognizing “woke up” levels. In addition, method 1 took 32.5 minutes to run, whereas method 2 took only 69 milliseconds to run, so method 2 had a huge increase in the runtime performance.

Clearly, the overall accuracy of the classifier still needs to be improved in order to ensure that we are accurately recognizing the sleep/wake patterns of the user, which I will discuss in more detail shortly.

One of the main limitations of my classification approach is in the size of the data in each 20-minute time bin of movement events. Each 20-minute time bin of movement events has 600 data points. In order to accurately quantify the similarity measurement between two different time bins, which is crucial for the k-NN algorithm, we cannot use an algorithm like dynamic time warping due to its quadratic runtime. This clearly has an effect on the scalability of my classification approach as the number of training 20-minute time bins and corresponding labels will need to be greatly increased in order to ensure a large enough distribution of user sleep patterns.

In addition, the user will want to see their sleep patterns with each predicted labels in real-time, which is currently not possible giving the dynamic time warp algorithm. This means that future algorithms will need to be very fast in recognizing sleep labels and the algorithm used to define the similarity metric between time bins needs to compress the 600 data points into a smaller number of features in order to ensure a fast, but also accurate algorithm for similarity metric calculation. In addition, I am currently running all of the movement event detection and classification algorithms locally using python scripts and fetching the needed data from the server. In order for the user to be able to view their sleep/wake patterns in real-time, I will need to move all of the processing for movement detection and classification onto the server side.

Conclusion and short-term plans

So far, I have successfully implemented an algorithm that reliably detects movement events. In addition, I have implemented a kNN classifier for recognizing sleep labels and tested two different methods for determining the similarity measurement between two time series of movement events. I have discovered that the kNN classifier using 4 key metrics is more accurate than kNN classifier using dynamic time warping and is significantly faster. Using the former method my classifier can detect user labels with 59.32% accuracy using a training and test set comprising six users and 16 nights worth of acceleration data and user-generated labels.

For milestone 4, my main goal is to improve the accuracy of my classifier in recognizing user labels based off of movement event data. I will look into the periodicity of different user’s sleep/wake behavior in order to see if I can improve the classifier using individual user’s typical sleep patterns. In addition, we can improve the classifier by enforcing spatial constraints due to the definitions of “restless”, “woke up” and “asleep” and typical user behavior, i.e. a user must previously have been “asleep” in order to “wake up”. Currently, I am using kNN, so I will continue to improve the method for calculating similarity measurement between two time series, but I will also look into other classification algorithms.

Secondly, I will need to move all of the processing work that is currently being done locally and manually onto the server. From this, the user will be able to see their sleep/wake patterns with our predicted labels automatically once they wake up. In addition, they will be able to see key metrics regarding their sleep for the night and how it/if it differs from their usual sleep behavior.

Once I have accomplished both of the tasks that I just mentioned the application would easily allow users to record their sleep and accurately identify their sleep/wake patterns. To the best of my knowledge, no other sleep-recording app combines data across users in order to predict sleep. By leveraging the network of users using the application, I will be able to provide accurate metrics that will help users understand their sleep and gather valuable data to help understand the key trends in human sleep.

REFERENCES

[1] Crean, Dan. *Sleepdex*. 27 April, 2014. <http://www.sleepdex.org/about.htm>

[2] Hong, Charles Chong-Hwa, et al. “REM sleep movement counts correlate with visual imagery in dreaming: A pilot study”.

[3] Hoque, Enamul, et. al “Monitoring Body Positions and Movements During Sleep using WISPs”. <http://www.cs.virginia.edu/~stankovic/psfiles/sleep.pdf>

[4] Owens, Judith, et al. “Television-viewing Habits and Sleep Disturbance in School Children.” *Pediat rics* Vol. 104 No. 3 (1999). American Academy of Pediatrics. 2013<http://pediatrics.aappublications.org/content/104/3/e27.full>

[5] Minnaar, Alex. “Time Series Classification and Clustering” <http://nbviewer.ipython.org/github/alexminaar/time-series-classification-and-clustering/blob/master/Time Series Classification and Clustering.ipynb >