Sensor Networks for Social Networks

by

Michael P. Farry

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

February 2006

Author [illegible]
Department of Electrical Engineering and Computer Science
February 3, 2006

Certified by [illegible]
Samuel Madden
Thesis Supervisor

Accepted by [illegible]
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Sensor Networks for Social Networks

by

Michael P. Farry

Submitted to the
Department of Electrical Engineering and Computer Science

February 3, 2006

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis outlines the development of software that makes use of Bayesian belief networks and signal processing techniques to make meaningful inferences about real-world phenomena using data obtained from sensor networks. The effectiveness of the software is validated by applying it to the problem of detecting face-to-face social interactions between groups of people, given data readings from sensors that record light, temperature, acceleration, sound, and proximity. This application represents a novel method for social network construction which is potentially more accurate and less intrusive than traditional methods, but also more meaningful than newer methods that analyze digitally mediated communication.

Thesis Supervisor: Samuel Madden
Title: Assistant Professor, Department of Electrical Engineering and Computer Science

Acknowledgements

I would like to acknowledge Charles River Analytics, Inc. of Cambridge, MA for sponsoring this work, and a few people within CRA for their help and ideas. I thank Zach Cox for sharing his work and insight into Bayesian networks, and Dr. Partha Kanjilal for his help in signal processing. I would like to thank Dr. Jonathan Pfautz, also of CRA, for his sound and insightful advice on this work and for introducing me to the topic of social network analysis. Finally, I thank my advisor, Prof. Sam Madden, for his thoughtful guidance and encouragement as this work progressed.

I. Introduction

This research describes a method to perform Bayesian inference on data traces generated by sensor networks. In this thesis, the method is used to monitor face-to-face interpersonal interactions. Signal processing techniques are used to measure the similarity between data streams over a time window, and Bayesian inference is used to estimate the likelihood that certain individuals are interacting.

Sensor networks can provide rich streams of data, but along with a huge quantity of data comes the potential for overload. Human analysts can take data streams and use tools to graph and visualize them, but as sensor networks scale over size and time, human review of data streams becomes impossible. The goal of this research is to take a large data set received from a sensor network, analyze it using signal processing, and use interesting features from the data to make meaningful inferences about real-world events. This technology is applied to monitoring individuals in an attempt to capture their interpersonal interactions. The TinyDB sensor network query system [2] was used to collect light, temperature, acceleration, sound, and proximity data from Berkeley motes [1]. Then signal processing techniques and *Bayesian belief networks* (BBNs) were employed to classify those sensor readings. Berkeley motes were selected because they are small, sample at relatively high rates, use small amounts of power, and are robust, low-maintenance devices. The current generation of Mica motes can be attached to an individual unobtrusively, allowing them to interact naturally with others. The next generation of motes will be tiny compared to the current ones, hindering individuals even less [3].

Social network construction and analysis allows us to gain a deeper understanding of human relationships. In a standard social network, people are represented as nodes in a graph, and the relations between people are represented as edges. Social network analysis can identify communication paths in places where effective communication is a priority, such as businesses [4]. Analysts can identify patterns of ineffective communication and make suggestions and improvements to operating procedures and organizational structure.

Before the advent of digital communication, the construction of social networks was done by hand: individuals were interviewed to determine who they know and interact with [15]. Such "analog" methods of social network construction were often tedious and inaccurate, since research has shown that people cannot accurately identify all of their complex social relationships [15]. Analysis of digital communication, such as e-mail traffic, gives analysts a larger, less biased data set to work with. However such analysis truly only analyzes a group's e-mail usage patterns, which may or may not correlate well to their actual interactive behavior. In contrast, the use of sensor network technology makes the automatic observation and characterization of all of a person's *face-to-face* interactions possible.

Through the use of scripted interactions performed by different actors, data sets were collected that represent a wide variety of human behaviors as they were performed by a variety of individuals. The resulting sensor readings between two individuals are compared using a variety of similarity metrics, and the metrics in turn are used as inputs to a Bayesian belief network. The training algorithms also account for the dynamic nature of real-world systems. For instance, the current state of interaction between two people is clearly dependent on the past state of interactions between them.

The software was designed to perform inference both offline and in real time. Some applications of the described methods might be useful on data collected and then analyzed at some future point in time, particularly if the data set needs to be filtered or otherwise processed by hand once it is collected. However, real-time inference makes the system more interactive: to determine the effectiveness of their models, analysts can see which real-world events correspond to certain inferences. Once a model is sufficiently trained, its observations are available instantly. Of course, the notion of live inferencing places a requirement on the algorithms used to perform the inference: they must run in a relatively short amount of time.

The method of data processing outlined in this thesis involves a number of steps. First, individuals participate in a scripted interaction session while wearing sensors, and data sets are collected. Next, the data sets are cleaned, filtered, and processed to prepare them for use in the Bayesian model. Then, machine learning algorithms are used to infer intervals of interaction. Finally, the model's inferential power is validated using test data and flaws in the model are revealed.

This thesis represents a research and development effort resulting in several contributions: (a) the application of sensor network technology to the domain of social networking; (b) the introduction of Bayesian belief networks to map sensor data traces to corresponding real-world events; (c) the use of signal processing techniques to identify intervals of signal similarity; and (d) the visualization of the construction of a social network as it is formed through individuals' face-to-face interactions. In summary, TinyDB was modified to utilize a suite of similarity metrics and signal transform tools on live streams of data, provide input to and receive output from a Bayesian network, and visualize the results. Bayesian network structures were developed to model the problem of identification of human interaction, and training methods for those networks were developed, implemented, and evaluated.

This document contains five Sections. Section 1 introduces the problem domain and provides a brief overview of the research that was conducted. Section 2 provides background knowledge about previous efforts in the fields of sensor networks, social networks, machine learning, and signal processing. Section 3 details the architecture of the system and the decisions that were made during design. Section 4 describes the techniques used to validate the methods, and Section 5 concludes.

II. Background

Previous work in a number of different fields is relevant to this thesis. Section II.A describes sensor networks, the tools used to extract data from them, and lists a number of projects that already employ sensor network technology. Section II.B introduces the concepts of social network construction and analysis, and explains how the monitoring of individuals' face-to-face interactions is useful. Section II.C explains Bayesian Belief Networks and Bayes' Law, and lists other machine learning techniques that have been employed when using sensor networks. Section II.D describes signal processing techniques that our tool uses to extract meaningful features from signals.

II.A. Sensor Networks

Berkeley Mica motes [1] are one of many versatile sensor node hardware platforms. Developed with the concept of power management as a priority, a Mica mote's 4 MHz processor consumes 8 mA of power when performing computation, and 15 μA when asleep. The two-way radio component of the mote, which transfers data that are collected, consumes a similarly parsimonious amount of power. Different sensor boards can be attached to the mote itself to record attributes such as light, sound, temperature, acceleration, and magnetic field. These motes utilize an operating system called TinyOS [5], and a programming language called nesC [6]. Programs can be structured conservatively with regard to power usage, giving the motes a battery life of up to a year for certain applications. Currently, the size of a rectangular mote is similar to the size of its battery pack, which holds 2 AA batteries. The next generation of mote

[3] will be many times smaller, measuring around 2mm x 2.5mm. Figure 1 displays the current and next generation of Berkeley motes. The next-generation device is the small square laying on top of the largest chip.

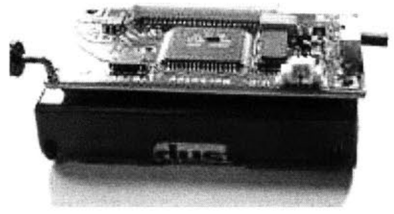


Figure 1. Current and next-generation motes.

Software abstractions have been written to make the usage of motes and TinyOS much easier. TinyDB [2] enables users to query a sensor network as if it were a database, using SQL queries. Users specify the attributes to be collected and the sampling interval, and the readings are reported to a base station. At the base station, the data readings are displayed in real-time, graphed, and logged if the user so chooses. The TinyDB/Mica mote combination is just one example of a sensor platform and its associated tools.

With such a powerful array of tools at their disposal, researchers have already applied sensor network technology to many fields. Gourley [7] describes the use of unattended ground sensors that record seismic, acoustic, infrared, and magnetic data in a military environment to locate and differentiate personnel and wheeled or tracked vehicles. Mainwaring et al. [8] apply sensor networks to the problem of habitat monitoring. Mica motes equipped with sensor boards known as "weather boards" monitored the conditions in nesting burrows used by a bird known as the Leach's Storm Petrel. The use of sensor network technology allowed the researchers to gain information about these birds without disrupting their natural habitat. Kumagai et al. [9] used a similar technique, spreading sensors over a meadow to observe a rare species of bird that they knew little about. Huang et al. [10] utilized sensors to monitor plant life to ensure that they received the proper light, temperature, and humidity.

II.B. Social Networks

Social network analysis involves the use of graph theory and other techniques to draw meaningful conclusions about people and the relationships among them [11]. The first social network graph was a "sociogram" created by Jacob Moreno in 1930 [12], and the research of Stanley Milgram in 1967 [13] produced the conclusion that two random United States citizens were linked by an average of six social network links. The growing field of social network analysis has gained particular importance with the recent explosion of digitally constructed social networks. By studying a data-rich social network, an analyst can draw many conclusions about the people involved and the ways in which they interact. Applications of social network technology include organizational psychology, the identification of communication paths, and military intelligence [14].

Constructing a meaningful, detailed social network is challenging. More traditional methods of social network construction involve interviewing subjects and recording their responses about who they know, how they know them, the nature of their relationship, and the nature of their communication [15]. These data collection techniques are vulnerable to human error and perspective. People may not be able to accurately identify all of the members of their social network, and two different people may describe a relationship between them in a completely different manner. An additional challenge arises from the fact that technological advances and social practices have diversified the means by which individuals communicate. Individuals may communicate through face-to-face interaction, paper ("snail") mail, telephone, e-mail, instant messaging, and other mediums. Recently, digitally mediated communication has been a focus of social network research because it leaves a distinct trace which can be parsed rapidly. Actual analysis of those traces may reveal a more accurate representation of a person's social network than simple interviews. The teams behind the commercial products InFlow [16], Spoke [17], and Visible Path [18], as well as research performed by Tyler et al. [19], attempt to identify key contact individuals in an organization through the analysis of individuals' email inboxes and other information systems traffic.

Steps have been made to allow individuals to manually construct and maintain their own personal digital social networks over the World Wide Web. The purpose of these sites ranges from informally providing personal information to friends to creating strategic social contacts for business purposes. The Facebook [20] allows college students to network and communicate with their classmates, while Myspace [21] allows anyone to create an online profile and create links to friends' profiles. Other sites such as Friendster [22] and Orkut [23] are centered around finding friends or potential significant others with similar interests. LinkedIn [24] allows professionals to find other people to seek for advice or collaborate with on business ventures.

The exact model used in a social network becomes important when considering the data that can be mined from it. Hence the structure of a social network is dictated, to a certain extent, by its purpose. One example of a social network whose structure is dictated by available data is the Link Analysis Workbench (LAW) [25]. The developers of LAW have sought to frame social network structures to a massive amount of unstructured data, as bits of military intelligence do not necessarily fit well into a traditional directed graph. The LAW's model is quite sophisticated. At the most basic level, one can imagine that a social network would be constructed with nodes representing people and edges representing relationships between them, but a more advanced representation may provide an analyst with better information. In the LAW, people, organizations, and events are all represented as nodes, to provide analysts with a better sense of how people are connected. Freeman et al. [26] seek to uncover organizational hierarchies through the analysis of asymmetric relations, and the structure of their social networks incorporates many one-directional relationships between people that reflect dominance, power, and influence.

The analysis of a well-constructed social network allows analysts to draw non-obvious conclusions and infer future behaviors or events. Friedkin [27] discusses the formation of social norms and analyzes their spread through a social network, which serves as a good example of how social network analysis can lead to insight on the psychology of groups. Jensen and Neville [28] use data from the Internet Movie Database [29] to construct a social network and predict a new film's box office success based on the participants, producers, and studios involved. Hauck et al. [30] have developed COPLINK, a tool which alleviates the data overload experienced by investigators when looking into a crime. COPLINK allows investigators to reason about which pieces of evidence are most important, and make inferences based on those pieces of evidence to provide conclusions which may not be readily apparent.

II.C. Machine Learning Using Uncertain Data

There are many machine learning and artifical intelligence techniques that have been developed. The major use of machine learning in this thesis is to take features from sensor data streams as input, and infer the state of interaction between people as output. As there are many possible sources of uncertainty that arise from that process, this section will discuss a machine learning technique called a *Bayesian belief network* that allows for probabilistic reasoning and the management of that uncertainty.

The handling of uncertain knowledge requires a system that makes rational decisions based on probability and utility [31]. A decision-theoretic agent must maintain a state of beliefs about the current state of the model, and be able to produce a rational outcome based on those states. A proper agent must also take into account the relationships between beliefs, and determine which variables are dependent on which others. A number of techniques have been developed that allow users to reason about uncertainty [32, 33]. These techniques include Dempster-Shafer belief functions [31], first-order logic [34], argumentation [35], and fuzzy set theory [36].

Bayesian belief networks [37, 38] utilize Bayes' Law, shown in Equation 1, to manage uncertainty. It states that the conditional probability of an event A occurring given event B is dependent on the conditional probability of event B occurring given event A, the *a priori* probability of event A occurring, and the *a priori* probability of event B occurring. As a result, Bayes' Law allows agents to combine evidence by computing an unknown probability from other, known conditional probabilities.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \qquad \text{Eq. 1}$$

A Bayesian belief network (BBN) is a data structure that utilizes Bayes' Law and a set of nodes that represent probability information to draw inferences. Each edge in the network is directed and is indicative of a parent-child relation between nodes. Each node contains a separate conditional probability distribution (CPD) that specifically and numerically relates its current state to the state of its parents. When an observation is made, evidence is posted to a node A. The effect of the newly posted evidence is calculated using Bayes' Law and reflected on another node B if A is in B's *Markov blanket*. Node A is considered to be in the Markov blanket of node B if A is a child or parent of B, or if A and B are both parents of a third node C. It should be noted that a node's Markov blanket effectively "shields" that node from the rest of the network; evidence posted to a node outside the Markov blanket will have no effect on the node's state.

BBNs and similar data structures have already been used to classify human activities. Brunette et al. [39] utilize Hidden Markov Models (HMMs) in combination with sensor network technology to predict a human's current activity. HMMs are a variation on the canonical BBN as described above. They are dynamic Bayesian networks (DBNs), which take into account the model's previous state when modeling the present [40]. HMMs model a Markov process, and contain "hidden" parent nodes along with observable child nodes. Actual state transitions occur within the hidden portion of the model, and observable data are then "emitted" with specific probabilities from the hidden states. Inferences can then be made about the hidden state using the emitted observable data. Brand et al. [41] use HMMs in vision tasks to classify two-handed actions. Bao and Intille [42] utilize classifiers of correlation coefficients, mean, energy, and frequency-domain entropy of accelerometer data to classify a person's activity. Lukowicz et al. [43] use HMMs and accelerometer and sound data to classify which of a variety of wood shop tasks a person may be performing.

II.D. Data Cleaning and Extraction Techniques

Sensor networks can provide a rich stream of data, but feature extraction is necessary for the Bayesian modeling described in this thesis. There are a staggering number of signal processing techniques that are potentially applicable to the problem of feature extraction from a data trace. This section discusses techniques that are applicable to two major tasks within this problem domain: filtering and similarity analysis.

Filtering is useful because noise is introduced to sensor data from a variety of sources. Two potential problems that can be mitigated by filtering are the observation of extraneous phenomena and the unreliability of sensor hardware. Transforms such as the Fourier transform and Walsh-Hadamard transform take data series in the time domain and transform them into the frequency domain [44]. Once the data are in that form, signals of specific frequencies can easily be filtered out. For instance, low-pass filtering [44] can be performed, filtering out portions of the signal that have a high frequency. This technique has the effect of smoothing out areas that appear to be erratic and noisy when viewed in the time domain. It is potentially useful for feature extraction, since it will filter out smaller, less significant or noisy events, leaving more important ones more easily visible. Once the major features are identified, they can then be mapped to particular events that occurred in the environment.

Similarity metrics are also useful for feature extraction. Numerical methods can be applied to two signals to generate some score that represents how similar the two signals are. If the signals are similar, there is a good chance the phenomena observed by the sensors are similar as well. Here is a list of potentially useful similarity metrics used in this thesis:

- The *Pearson correlation coefficient*, which quantifies the correlation of two random variables X and Y. Perfectly positively correlated data sets yield a coefficient of 1.0, while completely uncorrelated data sets yield a coefficient of 0.0.
- *Mean-squared error*, which computes the sum of the squares of the pairwise differences between points in the two signals. If most points in the data sets are close to each other, the resulting mean-squared error is relatively low.
- The *Cauchy-Schwartz inequality*, which is used in a signal processing filter known as the match filter. The inequality states that the magnitude of the inner product of two vectors is less than or equal to the product of the magnitudes of the vectors. The inequality is maximized (i.e., close to equality) when the vectors are linearly dependent. If two signals are linearly dependent, it is likely that they are similar. Thus, we can compute the absolute value of the inner product of the two signals. If that value is large, we can assert that the signals are similar.

- The *approximate entropy*, or randomness, of a signal. Comparing two signals' approximate entropy over a time window can give some clue as to their similarity. Unlike the correlation coefficient or mean-squared error, the individual values between two data sets are not compared. A data stream's entropy is calculated over a window, and compared to the entropy of the other stream.
- The *integral of the area* under the "curve" generated by a data trace. We hoped to quantify similarity by identifying signals with similar area values.

III. System Architecture

The goal of this research was to develop generic methods to take data traces from a sensor network, perform pairwise data similarity metrics, and utilize BBNs to create inferences about real-world phenomena. These methods were used to construct an application where sensors were attached to human participants to determine intervals of interpersonal interaction. An inference model was constructed that learned characteristics of human interactions using data from a network of sensors. The model was then used to perform real-time inference about which individuals were interacting with which others.

Specific methods for data collection and analysis, feature extraction, machine learning and BBN structuring, learning, and inference were developed. A number of challenges arose from the unique characteristics of the system, including: the large amount of background noise present in the data collection, and the variability between individuals and between environments. Section III.A describes the methods used to collect the data and the challenges faced in adapting the Berkeley mote network to the problem of monitoring interpersonal interactions. Section III.B explains the methods used to translate the stream of numerical data provided by the sensors into a format that could be used by a BBN. Section III.C outlines the chosen structure of the BBN that was used to perform inference, and argues the correctness of that structure. The learning algorithms that are used to train the system are explained in Section III.D, while the methods used for inference follow in Section III.E. Section III.F concludes with details of some of the scalability concerns that may be encountered as the system grows.

III.A. Data Collection and Cleaning

One fundamental question to be answered was where to place the motes to collect the most meaningful data. The Berkeley mote hardware was not designed with person-monitoring capability in mind. As a result, a few adaptations had to be made to accommodate the difficulties that arose in monitoring individuals. For the purposes of data collection, each participant was assigned one mote. The mote was clipped to his or her shirt collar, in an attempt to get the microphone sensor close to his or her mouth. No additional motes were used to monitor the environment. However, in the event that a data collection experiment took place over a large space, additional stationary motes were distributed to help with the logging of packets.

The Berkeley mote network can be assigned to sample at different rates. The desire to obtain as much data as possible was balanced with the demand for reliability. If the sampling rate was pushed too high, too many packets were pushed through the network, and data were dropped, or the base station lagged due to data overload. As a result of a few tests, it was determined that a sampling interval of 512ms was optimal. A network of seven motes sampling at 512ms reported smoothly, even as the motes spread out over a distance and routing became more complicated. Pushing the sampling interval down to 256ms resulted in frequent data loss. It should be noted that with a sampling interval of 512 ms, there are small events that may go undetected.

In addition to the standard TinyDB software package, TASK (Tiny Application Sensor Kit) [45] was used for data collection. TASK is a toolkit that supports the idea of a "Sensor Network in a Box," and requires less sophistication on the part of the user. As such, TASK removed some of the biggest hurdles for repeated data collection and logging. Specifically, TASK was used to query the sensor network and log the resulting data sets in a PostgreSQL database [46].

The fact that people were moving while wearing the motes introduced some challenges. Preliminary testing was done before any formal data collection to confront these challenges. Obviously, if a person moved out of any other mote's range, the corresponding data would not reach the base station. However, a mote that moved around very much also dropped a small but significant portion of its data. Also, there were some points in data collection where a mote would stop reporting for some unknown reason. There needed to be a way to distinguish a person who was legitimately out of range from a person whose mote was malfunctioning or who was simply walking around. These "data holes" are dealt with differently in learning and inference, and the methods for accommodating them will be described in those sections.

A proximity reading, measuring each mote's physical distance from every other mote, was created. Every epoch, each mote recorded the node id number of each other mote it heard. As a node pair's proximity decreased, it was less likely that they heard each other. The proximity reading of this system for nodes X and Y, is the number of epochs where either X heard Y or Y heard X over the last Z epochs, divided by Z. When the nodes are far away, the proximity is zero. When they are close, the proximity approaches one. The proximity metric is dependent on other factors, including line-of-sight considerations and the total number of nodes in the network. If there are more nodes in the network, there are more packets in flight and it is less likely that X and Y will hear each other.

Another challenge was the development of a mechanism to register and record the interactions as they happened. Having participants somehow flag their interaction state was too complex, because they would have to keep track of whether or not they were interacting with every other participant. That would have been a heavy cognitive load, and a burden that might interfere with a participant's natural communication with others. It was determined that the best way to tag the sensor data with intervals of interaction was simply to videotape the data collection session, then mark the data by hand later.

In summary, the typical data collection methods used for a Berkeley mote network needed adjustment to fit the problem of monitoring interpersonal interactions. To address these concerns, we placed one sensor on each individual and added tools to deal with missing data and measure proximity.

III.B. Data Feature Extraction and Processing

To determine the events of interest that would clue the system in on the individuals' interactions, a number of techniques were employed to extract interesting features from the data streams. The noise picked up by the sensors and the unreliability of the hardware made this problem more difficult. The system needed to be able to distinguish between events that signaled an interaction (such as two tagged individuals speaking to each other), and events that were inconsequential (such as a bird chirping outside).

For the purposes of learning, small holes in the data are filled in to provide the training algorithms with a more solid data trace. Dropped packets register as a value of zero for each attribute, and legitimate values of zero are never given for any attribute. If a hole in the data lasts for fewer than five seconds, then the values on either side of the hole are averaged, and the average value gets filled in for each part of the hole. Five seconds was chosen to be the ideal interval for data interpolation; if a data set was missing for more than five seconds, it was reasonable to presume that the interpolated data would not be valid, as significant events would likely occur. If the hole is longer than five seconds, it is considered to be legitimately missing data and stays at zero. TinyDB gives each piece of incoming data a sequence number. Once dropped data are accounted for, the sequence numbers are renumbered in each sensor's data stream so they are perfectly synchronized. Due to the high sampling rate and other factors, sequence number X on one sensor's data log did not necessarily match up to sequence number X on another sensor's data log. Instead, the time of the arrival of the data at the base station was used as a more accurate indication of relative samples. Once the sequence numbers were properly reconciled, the data were marked by hand to indicate intervals of interaction, to facilitate learning during the training phase.

Every comparison drawn between data streams is done in a pairwise fashion, separately for each attribute. To determine if X is interacting with Y, we compare the data streams of X and Y directly, not involving any other sensor's data. Of particular interest is a measure of the similarity of the two signals over a specified window of time. In an attempt to provide a diverse array of techniques for experimentation, a number of metrics were implemented, which are listed in the Background section. The motivation behind the development of similarity metrics was to identify points in time where signals are similar, and thus more likely to reflect an interaction between people. One fundamental hypothesis used in this work is that signal similarity indicates similar surroundings, and thus a higher probability of interaction. For instance, if one person's sound signal is relatively flat and another's shows activity, we hypothesize that they are likely to be hearing the sounds of two different environments, and are not likely to be interacting. A similarity metric will indicate that the signals are dissimilar, and help make the correct inference.

Conditional probability tables (CPTs) were used in the BBN structure, so inputs needed to be discretized. When the data are used for learning, the similarity metrics are determined at each epoch for each attribute. The values are then sorted and equally-sized bins are created. Each value is then assigned a bin. In addition, "missing" data epochs register in a separate bin. The reasoning for this will be described in the next section, which is a discussion of BBN structure.

This part describes data pre-processing that was done to alter sensor data traces into data that could be used by a BBN. Methods for data interpolation, identifying intervals of signal similarity, and data discretization are discussed.

III.C. BBN Structure

BBNs were used to make probabilistic inferences about the state of interpersonal interactions between people, given evidence provided by the sensor network. Nodes in the BBN utilize conditional probability distributions, which can help manage the uncertainty inherent in a problem like this one. The BBN structure described in this part, like the similarity metrics described in Part B, compares two sensors' readings in a pairwise manner. The BBN makes a prediction about whether the pair is interacting based on the values of the similarity metrics, as well as its own guess as to whether or not that pair was interacting over the last epoch. BBNs were created, trained, and run with Charles River Analytics' BNet toolkit, a Java software tool [47].

A pairwise analysis was chosen over other possibilities due to its simplicity. Another possible model would consider an interaction over a certain period in time, and attempt to figure out which sensors' data matched over that period in time, thus giving a list of participants in that interaction. This structure would involve many nodes, complicated probability distributions, and possibly changing the structure of the BBN as nodes are "added" and "removed" from the interaction. The predictive power of the pairwise analysis has proven effective, so there is no immediate need to develop a more complicated structure. The structure of the BBN was selected rather than learned, and fully observable nodes were used (similarity metrics for each attribute, and whether or not the pair was interacting) to simplify learning. To learn a fully observable network with a known structure, simple frequency counting is all that is needed [40]. The use of a partially observable network requires the use of the EM-eta learning algorithm [40], and not specifying a certain structure requires the use of structural inference algorithms. There is merit to these alternatives, and they are discussed more fully in the Future Work section.

One main question with regard to the selection of structure is the identity of the nodes. The BBN structure contains a sensor pair's similarity metric value for each attribute, a proximity indicator, an "Interaction?" node with boolean states, and nodes containing the state of the system in the last epoch. Other alternatives existed, since certain sensor readings do not necessarily correlate to interpersonal interactions. For instance, certain "secondary" conclusions could be deduced, and those could in turn be used to conclude whether an interaction was occurring. Examples of such secondary conclusions would be whether or not the two individuals were speaking at all, whether or not they were speaking to each other, whether or not they were in the same room, and whether or not they were in motion. More advanced and sophisticated models are a subject for future work.

Once the nodes were in place, the edge structure also needed to be developed. Bayesian networks are a cause-and-effect model, where Event A causes Event B to occur with some probability. Two different possible structures for the BBN were selected, based on two types of reasoning: *abductive* reasoning and *deductive* reasoning [48] . Deductive reasoning takes evidence that is present and attempts to draw a meaningful conclusion from it. In the BBN generated by deductive reasoning, the edges are directed from each similarity metric to the "Interaction?" node. Its structure is illustrated in Figure 2.

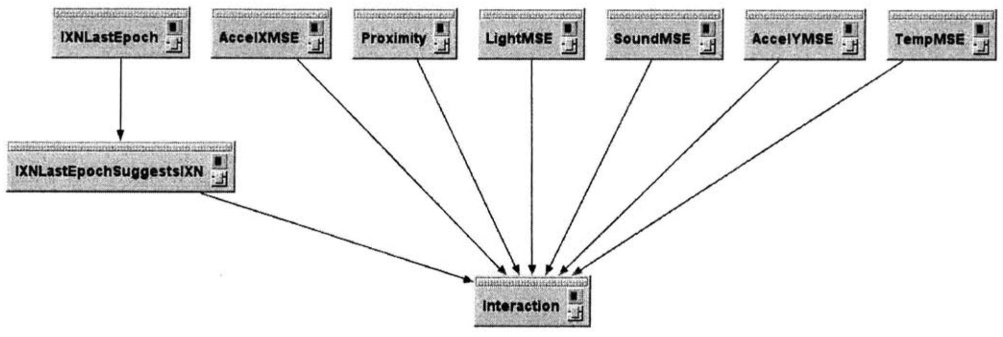


Figure 2. A BBN constructed with deductive reasoning.

The line of reasoning in the deductive case goes as follows: "I see these sensor readings, and they cause me to believe there was an interaction. The consequence of the sensor readings is my belief that there was an interaction." The sensor readings cause the belief, and the direction of the arrows reflects that reasoning. The line of reasoning in the abductive case is: "I see these sensor readings. Why did they occur? Because of the presence or absence of an interaction." In this case, each edge points from the "Interaction?" node out to the sensor reading generated by that interaction. The abductively reasoned BBN structure is displayed in Figure 3.

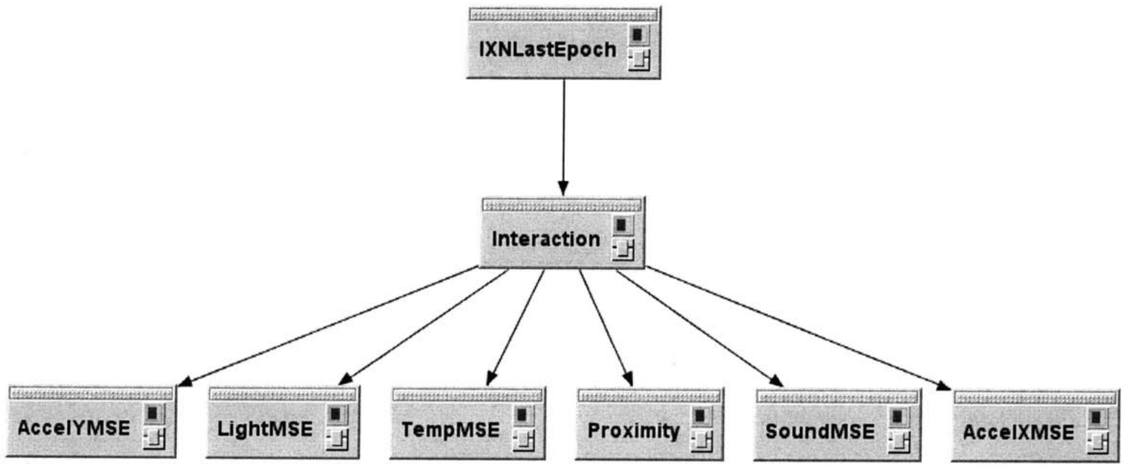


Figure 3. A BBN structure using abductive reasoning.

Early tests on one data collection session revealed that a BBN structure utilizing deductive reasoning performed slightly but noticably better than one using abductive reasoning. As a result, the final BBN structure utilizes deductive reasoning.

The similarity metrics' nodes' outputs all feed into the root node, called "Interaction." The conditional probability table (CPT) for that root node gives the probability that the two individuals under consideration are interacting for any given combination of similarity metric values. Each attribute registers in a given bin, and a probability of interaction is given as a result as seen in Equation 2.

$$P(Interaction \mid Attr1 = a, Attr2 = b, Attr3 = c) = x \qquad \text{Eq. 2}$$

If x is close to zero, we can be relatively certain that the individuals are not interacting. If x is close to one, we can be relatively certain that the individuals are interacting. If x is close to 0.5, then we cannot be certain about the outcome. A sample BBN structure with evidence posted and a corresponding CPT can be seen in Figures 4 and 5.

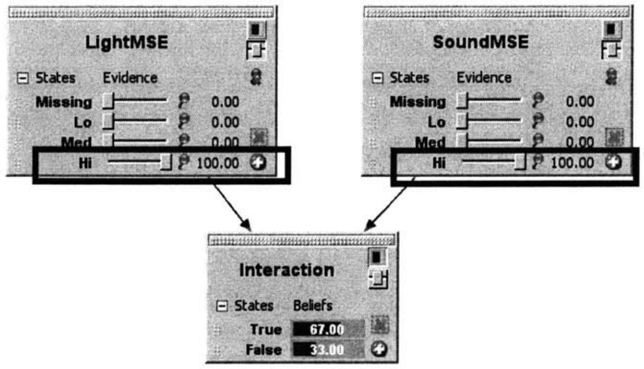
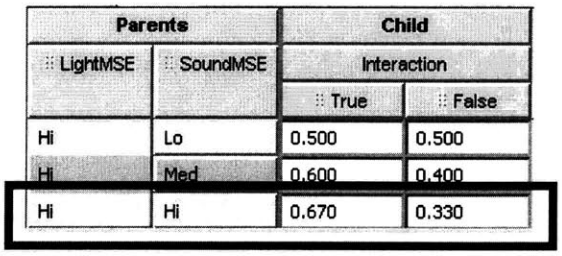


Figure 4. A sample BBN with evidence posted and a resulting inference.



Parents		Child	
LightMSE	SoundMSE	Interaction	
		True	False
Hi	Lo	0.500	0.500
Hi	Med	0.600	0.400
Hi	Hi	0.670	0.330

Figure 5. The conditional probability table that generated the inference shown in Figure 4.

If a pair of individuals is interacting at time *t,* it is likely that they are also interacting at time *(t+1).* For the scope of this thesis, an "interaction" between two people lasts for a reasonably long amount of time (i.e., short interactions where people pass each other in the hallway and greet each other are not considered.) The model's inference should reflect this consistency, and the duration of a predicted interaction should match the duration of an observed interaction. To enforce that, the previous inferred state of the system has a relatively large effect on the current inferred state of the system.

In a real-time predictive system, when a mote fails to report for a few epochs, it can be difficult to tell if the mote is legitimately missing (i.e. out of range) or simply not reporting. Looking strictly from the BBN's point of view, the one conclusion that can be drawn is that a missing data trace over a long period of time is very meaningful. If it is missing for a long period of time, the sensor that is not reporting can be presumed to be legitimately missing, and thus not interacting with anyone. This rule is not explicitly enforced in the structure of the BBN, but during training, the system quickly learned that an extended period of missing data meant no interaction.

In summary, this part outlines the considerations that led to the final BBN structure used in the system. The identification of real-world states and the nodes considered in the BBN are discussed, along with the decision to use deductive rather than abductive reasoning. The temporal nature of the data and structural decisions regarding that aspect are also covered.

III.D. Training

A very simple learning algorithm was employed for this system. An advantage of the BBN structure as described above is that each node corresponds to an observable value in a learning data set. Since we are dealing with a fully observable BBN with a known structure, learning simply involves frequency counting. For each combination of possible states for light, temperature, accel-x, accel-y, sound, proximity, and the state of the last epoch, an entry in the conditional probability table was created. The formulae used to calculate that entry are illustrated in Equations 3 and 4.

$$\textit{Event } E = (\textit{Light} = t, \textit{Temp} = u, \textit{AccelX} = v, \textit{AccelY} = w, \textit{Sound} = x, \textit{Prox} = y, \textit{IXNLastEpoch} = z) \qquad \text{Eq. 3}$$

$$P(\textit{Interaction} \mid E) = \frac{\sum_{i=0}^{n} E \wedge \textit{Interaction}}{\sum_{i=0}^{n} E} \qquad \text{Eq. 4}$$

By comparing the frequency with which a certain state correlates with an interaction, we can predict the probability of an interaction occurring when that state is observed later. In addition, *a priori* probabilities for each event were established using frequency counting.

III.E. Inference

Once the learning algorithms were developed, methods were outlined to infer interactions based on incoming data. The goal was to construct an inferential system that could operate in real-time, so the temporal nature of the data coming in drove the algorithms' design. As a result, the inferential methods differ quite substantially from the learning methods.

The inferential framework was constructed with TinyDB as its foundation. TinyDB already has the capacity to accept data from each queried sensor and organize them, so it was only natural to extend TinyDB's functionality to include inference. The inferential system keeps track of data reported by each sensor over the requested time window of ten seconds. The relevant pairwise similarity metrics are computed in real-time whenever a sensor reports new data. These metrics are then binned according to the learned bin values, and posted as evidence to the learned BBN. The BBN then produces a probability of interaction.

To aid in verification and help to visualize the inferences of the system more clearly, several visual aids were added to TinyDB. As distance metrics are calculated between each pair of sensors, they are graphed. As the system continually guesses that certain individuals are or are not interacting, a linked graph is displayed and updated to reflect the most current guesses. Each participant is represented as a node, and links between the nodes appear or disappear as interactions start or stop.

As with any machine learning system, this model must be trained on data that are similar to the data that will be collected during inference. For instance, if the system is trained in a uniformly low-light environment, it will be likely to learn lower bin values for the mean-squared error of light between a pair of sensors. If it is then used for inference in a brighter environment, or an environment with very variable lighting, every calculated mean-squared error value for light will fall into the highest bin. This confusion will skew the system's ability to draw meaningful inferences. Fortunately, however, it appears that the variability both between sensors and between the same sensor during multiple trials in similar environments is minimal, and the same skewing effect is not seen.

A considerable amount of computation is performed whenever a sensor reports. The time window is updated, and old data are replaced by new data. For a system with n sensors, $\sum_{i=0}^{n-1} i$ similarity metrics are re-computed every epoch for each attribute. Thus the same number of instances of new evidence are posted to the BBN, producing the probability that any given pair of sensors is currently involved in an interaction. The system is quite robust and is currently able to perform and visualize inference on incoming data from at least seven motes at the current sampling rate.

III.F. Scalability Concerns

The concern regarding system scalability is one drawback to the pairwise analysis. Scalability matters most in the case of live guessing, where data must be processed and inferences made as soon as possible. However, adding more sensors also makes tagging collected data, as well as learning the BBN models, more complicated. When a new sensor is added to the system, its interactions and data traces are linked to each other sensor in the system, and a combinatorial explosion results. Given n participants, the total number of potential interactions is equal to $\sum_{i=0}^{n-1} i$. If there is a need to observe many participants, the pairwise analysis may not work well. Even if the pairwise analysis is still used, other clever techniques can be employed to mitigate the effects of having many sensors. In a system with many sensors, an observer may consider only a few potential interactions interesting. A future system could be built with the capability for a user to mark only specified data traces for comparison. Such a system could effectively narrow its scope, eliminating superfluous computation.

IV. Procedure

To validate the utility of the methods presented in Section III, several relevant sessions of data collection were performed. Each data collection provided the impetus for several changes to the design of the system. Some of the changes were specific to that problem, but many reflected weaknesses in the overall design of the methods.

Once the most basic model was constructed, a script was drawn up for human participants to follow. Next, a data collection session was performed by the participants according to the script. Necessary data pre-processing was performed, and a BBN was trained on the gathered data. Proper validation involved splitting up the gathered data into a training set and a test set, and the inferences generated by the training set would be checked against the actual observed results in the test set. Based on the results, improvements to the system design were implemented, and a new script would be developed to test the improvements and investigate other potential faults in the system. The process is illustrated in Figure 6.

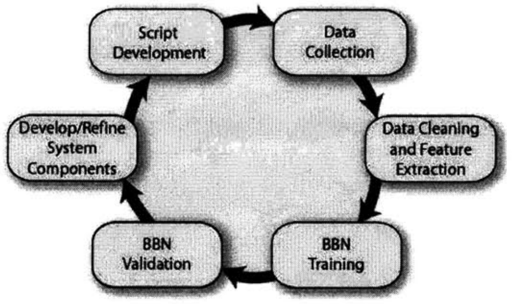

Script Development
Data Collection
Data Cleaning and Feature Extraction
BBN Training
BBN Validation
Develop/Refine System Components

Figure 6. The iterative design process for this work.

A. Validation Goals

The main goal of validation was to make significant incremental improvements to the model from session to session, identifying weaknesses in the methods used. The ideal model would enable perfectly accurate inference and classification of human interactions in real-time over a long period without regard to environment, background noise, interaction with untagged individuals, interaction type, duration of interaction, number of individuals participating in an interaction, or the number of separate interactions involving tagged individuals. That ideal model is unrealistic, but it serves as a clear outline for a perfect system. As the methods were refined and improved, the scripted constraints of each successive data collection session were loosened. The model was also designed to be generic and applicable to other situations, rather than specific for this problem domain.

There are several factors which make these goals non-trivial. Even when keeping the participants, script and environment constant, no two data collection sessions will yield the exact same data traces. The model must be able to handle a huge amount of variability. Over-fitting to a certain scenario would also undermine the generic nature of the methods. In consideration of those potential shortcomings, each script was constructed to provide a rich variety of interaction types over an extended period of time. The following algorithm was used to provide a numerical value for the correctness of a model:

```
For each sensor pair (I,J)
  Training set ⇐ Every pairing except (I,J)
  Train BBN using training set
  For each epoch
    Prediction ⇐ Inferred interaction state
      from BBN based on posted evidence from
      I and J
    Correct ⇐ Observed interaction state
    If (Prediction == Correct)
      CorrectEpochs ⇐ correctEpochs+1
  PercentCorrect ⇐ (CorrectEpochs / (number of epochs * number of pairs))
```

B. Script Design

Scripts for each data collection session were carefully designed to identify flaws in the inferencing model and drive improvements to its design. The most basic elements of a script were the cast and the content. Using a larger number of participants, the session would yield a larger variety of interaction types, but a "smaller" interaction would be drowned out by all the other data that were being collected. Also, from a technical standpoint, the number of participants involved actually had various effects on data collection. For instance, if a large number of participants participated, many packets would be transmitted from mote to mote, and the extra traffic would drive down observed levels of proximity.

The events dictated in the script had to be designed such that diverse types of interactions were represented for significant amounts of time. The observed data traces from a lecture, where one participant addressed an audience for a long period of time, differed greatly from the traces obtained from a lively round-table discussion. The sound attribute proved to be a good indicator of who was speaking. Relatively high levels of sound from one sensor only for a period of time indicated that one speaker was delivering a lecture, while high levels in shorter bursts from multiple sensors indicated a more lively discussion. Scripts were generated that allowed the observation of a rich array of activities, guiding participants to do such things as arriving in meeting areas together or separately, sitting or standing while interacting with others, and moving or remaining stationary as they spoke. Yet it was stressed to the participants that their actions should remain completely natural, and scripts were designed to ease an participant's cognitive load.

The content of the script dictated the learned *a priori* probability of interaction between two individuals. To properly balance the system, there needed to be large periods in which some participants are not interacting with anyone. Otherwise, the model would have been biased, and more likely to infer false interactions. Thus the *a priori* probability of interaction that led from a data collection script matched the expected probability of interaction in the guessing environment.

Each script was designed to give the system a closer look at interactions that it did not identify correctly in previous iterations. As a result, the types of interactions specified in the script generally became more and more specific. In Script 4, for instance, a large part of the script involved monitoring individuals who were in close proximity, but not interacting. A summary of the types of interaction outlined in each script is available in Table 1.

Script #	**# of Participants**	**Interaction Types**
1	4	70% Round-table discussion 30% Travelling in groups
2	7	50% Lecturing 30% Round-table discussion 20% Separate groups interacting in a shared space
3	4	75% Round-table discussion 25% Separate groups interacting in a shared space
4	3	65% Round-table discussion 35% Individuals in proximity but not interacting

Table 1. Summary of scripts used in data collection.

C. Data Collection Procedure

To conduct a data collection session, all participants were assembled and given their scripts, as well as some simple instructions. One mote was attached to each participant's shirt collar. With the mote in that particular place on every person, it is easy to distinguish a high-intensity period of sound as that person is speaking. Lower-intensity periods of sound can be identified as other people speaking, or background noise.

In addition to the participants, one person was present who was responsible for monitoring the process, recording the exact times that interactions occurred, and making sure collection went smoothly. For a few of the data collections, a video camera was set up to augment the director's observations. Later, using the video and the written records, the recorded data were augmented for the purposes of learning to provide information regarding interactions at each epoch.

As described above, the participants were instructed to interact naturally, but within the bounds of their script. The times specified in the script were not followed to the letter, but if necessary, the director would give cues. No scripts specified locations for the interactions to take place, nor did they specify with detail where individuals should be placed in a room. As a result, even when following the same script, environments, signal intensities, and intervals of interaction differed greatly. This condition drove the design of the system to be robust and handle variability.

D. Usage of Signal Processing and Feature Extraction

The signal processing techniques that are used in the final version of the system are described above in the System Architecture section. However, significant experimentation took place involving different signal processing techniques once data were collected, to visualize the data streams and identify features that might aid the inference of interactions. The lessons learned from simple inspection of the data traces had a great effect on the final design of the model.

MATLAB was used to process and graph the collected data. Specific patterns in the collected data were identified and found to be consistent from session to session. Some of them were quite surprising. For instance, analysis of the data trace of light on a sensor was actually a good indicator of a person's motion. Whenever participants would remain stationary, their light readings remained constant. When they would start to move around, even into similarly-lit areas, their light readings would jump around wildly. Figure 7 displays light levels for two sensors taken during the second data collection session. The data trace from Sensor A varies significantly from epochs 750-1500, corresponding to a point in which A was delivering a lecture, standing and moving around while talking. In contrast, Sensor B's signal is flatter for extended periods of time, when B was sitting down. Dramatic changes occur in B's light level at epochs 600 and 1400. These shifts correspond to a room change for B.

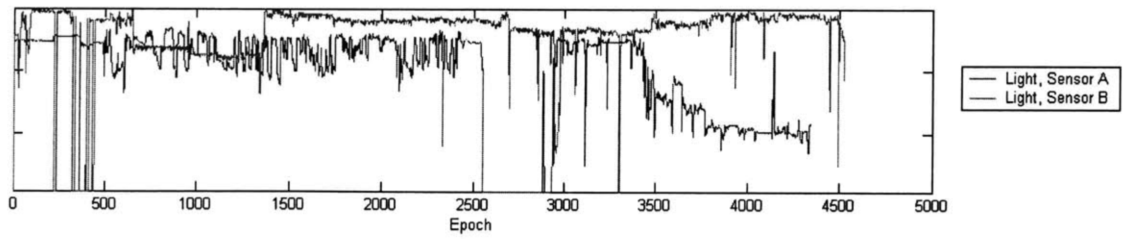


Figure 7. Light levels for two sensors captured during the second data collection session.

Similarly, the analysis of sound intensity is more difficult than originally thought. The sound signal is very erratic, and due to split-second differences in sampling, a specific sound may register completely differently on two sensors. The challenge of predicting interactions using sound inspired the use of many different signal processing metrics. The success of the mean-squared error in identifying periods of interaction given traces of sound data is a major reason why mean-squared error was selected as the primary similarity metric. However, each of the similarity metrics that was implemented is available for use, and switching from one metric to another is trivial. Mean-squared error was the most useful metric in this problem by a slight margin, but other problems may require different metrics. Figure 8 displays the sound data trace from a sensor that was exposed to the same person continually speaking over the entire interval, and clearly shows its erratic nature.

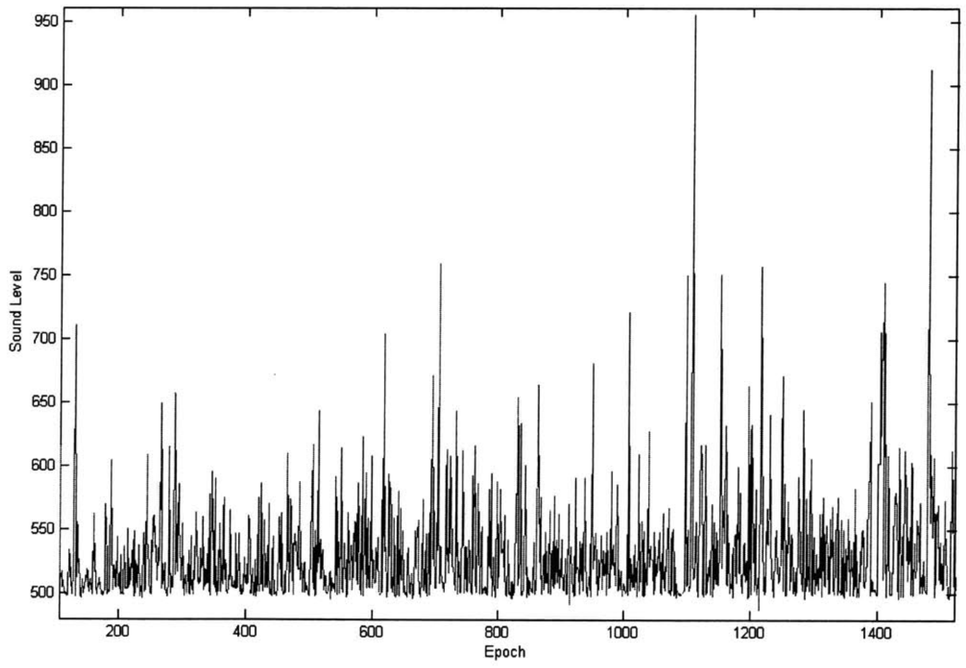


Figure 8. Sound recorded by a sensor that was exposed to an individual speaking over a period of time.

The difficulty in identifying sound signals suggested that filtering the signals might be useful. Specifically, low-pass and high-pass filters were used in MATLAB to "smooth out" the sound signals in both the test and training sets before validating. No filter seemed to have a positive effect on the system's predictive power. The filtering techniques were abandoned due to this lack of results, as well as the fact that live data coming in would be hard to filter. Performing a Fourier transform and filtering on a small window of data would provide less accurate results than using a whole session's data, but using all the data would not be practical in the live case. In addition, even if it worked well, the filtering would add more computational complexity, worsening scalability concerns. Despite its lack of usefulness in this problem, the filtering framework is still available for other potential applications. Table 2 demonstrates the percentage of epochs accurately identified in a trial using unfiltered data, high-pass and low-pass filters, filtered above and below specific frequencies.

Transform	Filter	Frequency	Accuracy
None	None	N/A	71.84%
Fourier	High-Pass	100	54.83%
Fourier	Low-Pass	25	57.47%

Table 2. Percentage of accurately identified epochs when using unfiltered data, high-pass and low-pass filters.

E. Validation

Once the data had been processed and the model had been trained, the model was validated. In the first three data collection sessions, the script was only performed once. In the final data collection session, the script was run twice. Different participants ran through the script in the same environment, so two different data collections really took place. Since there was only one set of data in the first three experiments, validation was performed by splitting up the data into test and training sets. Since inference was done in a pairwise fashion, validation was performed using the algorithm outlined in Part A.

In addition to a raw percentage of correct epochs, graphs were constructed that highlighted areas of incorrect inference. This feedback was the most useful in making system improvements; it quickly became clear what conditions "tricked" the model. Using this automated method of validation also allowed changes to be made, which could then be validated on the same data set. This method was useful in determining which BBN structure and similarity metrics provided the best accuracy.

For the last experiment, two data sets were collected, meant to simulate validation by actually learning a system, then performing real-time inference. However, running real-time inference more than once would require attendance by all participants. Instead, one data set was used for learning, using the normal data pre-processing before feeding it to the model. The other data set was not pre-processed, and was used for inference as if it was coming in real-time. Any data holes or other aberrations were not removed.

F. Domain-Specific Design Notes

Whenever these methods are applied to a specific problem domain, BBN structure, choice of similarity metrics, filtering, and other parameters will have to be fine-tuned to obtain optimal inferential performance. This section outlines the iterative process used to fine-tune the software to be used as a system for identifying face-to-face interactions. In-depth analysis of each data collection session gave very valuable information about the flaws of the model. The first data collection session was very constrained; there were not very many interaction types, and the participants stayed in the same space for the majority of the session. All speech in that session was done in a round-table discussion, and one participant left for only a short period of time. The model performed well, so it became clear to us that a more diverse script was needed.

The second session was a seven-person scripted interaction that involved a lecture, a round-table discussion, and participants breaking up into groups and walking and talking with each other afterwards. This session revealed a bias in training towards over-represented types of interaction. The lecture in the script took place for a long time, so the model was well trained to identify the individuals involved in the lecture. However, the portion of the script where the participants splintered into different groups lasted only a few minutes; due to this, the model could not infer the correct results. The development of the proximity attribute was inspired by this failure. It was hypothesized that the participants' close proximity to each other would enable the system to infer their interaction, and the next script was structured to test that hypothesis.

The third data collection session was simpler, and was essentially contructed to test out the proximity attribute. The results were encouraging. However, upon performing a sensitivity analysis [47] on the BBN, it appeared that the model was depending quite heavily on proximity and ignored other attributes for significant stretches of time. For the final data collection session, the script included stretches of time where two individuals were in close proximity, but were not interacting. The script simulates a scenario where two individuals who share an office are working, then are visited by a third individual. Once the third individual leaves, the officemates go back to work. The model's ability to infer these interactions reflects versatility that was built into the system through trial and error.

V. Results

The results of this work are encouraging; with fine-tuning, these methods can be used to accurately identify face-to-face interactions in relatively complex settings with a probability of success ranging from 82-95%. In the final data collection session, two data sets were collected using two different sets of participants following the same script. One data set was pre-processed and fed to the model as training data. The other data set was presented to the system as if it was being collected in real-time, and the model's inference was checked against real observations. The model correctly identified each of the six state transitions among the three actors. In addition, the model correctly classified 82.4% of the total epochs that were gathered. Figure 9 shows the actual intervals of interaction against the system's inferences for one particular pair of sensors.

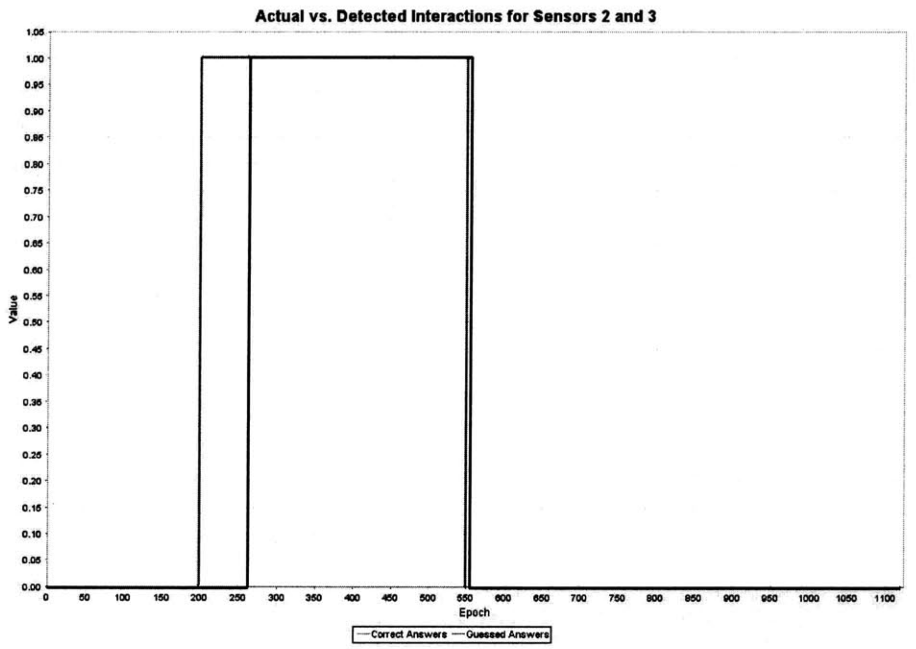


Figure 9. Actual vs. Detected Interactions for one particular pair of sensors. "High" values indicate epochs for which there was an interaction, while "low" values indicate a lack of interaction.

Many of the epochs that were wrongly identified suffer from a lag effect; the data collected that suggest a different state took a while to exert their influence on the similarity metrics, as well as the model's reliance on past inferences. In previous data collections, splitting of the data set into test and training sets yielded encouraging results as well. These results are summarized in Table 3.

Script Number	**Percentage Correct, Splitting Test/Training**	**Percentage Correct, Comparing Sessions**
1	86.44%	N/A
2	84.32%	N/A
3	94.91%	N/A
4	85.05%	82.40%

Table 3. Accuracy of Interaction Detection.

While the current results are encouraging, there are still some points of inaccuracy, and improvements can still be made to the model. The results of the final data collection sessions indicate that the model may still be over-fitted to cases where people are in proximity and interacting. In addition to that, other tests suggest that there is an over-fitting problem with regards to changes in environment. Using a model's learned values for binning the similarity metrics is wholly dependent on the relation of the learned values with the values that may be present in the new environment. Ideally, the system would be able to ignore any difference between environments as well as it currently seems to ignore differences between people. As it stands now, a user would be required to tweak bin values manually based on knowledge about the environment, but an automatic technique for learning or calibration would be ideal.

Several alternative models proved ineffective. Before using similarity metrics, the system binned raw signals from a pair of sensors, and took a measure of how often that combination of signal readings for those modalities indicated an interaction. For example, consider a new reading coming in where sensor A reads “Low” and sensor B reads “Medium”. The system selected all of the epochs in the training set that matched those readings, and compared how many were marked with an interaction against how many were marked with no interaction. That ratio dictated the probability that the selected modality “suggested” there was an interaction going on. Each modality’s “suggestion” was weighted and factored into a main “Interaction” node. This model is flawed because it is susceptible to differences between sensors and large changes from one epoch to the next. When measuring sound, for instance, the model was fooled because the sound values varied drastically. In contrast, the similarity metrics used in the current model are calculated over a specific time window, and provide an accurate sense as to whether two sensors are being exposed to the same environmental phenomena. The weighting system used in the root node is not optimal, either; the weights were tweaked by an algorithm that sought to produce the highest accuracy. The algorithm often produced nonsensical results; one on occasion, it produced a model that was completely dependent on light. When that particular model was tested with a different data set, its results were quite poor.

Models with different numbers of bins were tested, with the intuition that a greater number of bins would narrow the system’s focus and provide more accurate results. Over the course of a few trials with different data sets, however, it became clear that increasing the number of bins beyond four produced very small or even negative returns. Also, the CPT in the main "Interaction" node contains an entry for each possible combination of states. BBNs with a large number of bins had huge CPTs, and became unable to perform inference on live data traces. Furthermore, training a model with large CPTs was not only time-consuming, but also incomplete. There were unusual combinations of states that did not appear at all, or very infrequently, in the training data. When these states were encountered during inference, the system often made nonsensical assertions. Using a model with four bins ensures that all states are covered well within the bounds of the training data sets that were used.

The system is not specific to this problem domain. As it stands now, the system can take any number of data traces as a training data set, filter the training set, and apply any number of similarity metrics to the training set. A developer can then select or learn a BBN structure to use as a model with the training set, and train the BBN. BBN training in the fully observable case with known structure is fairly generic, and only a small amount of work would be required to learn a BBN whose structure is not significantly different from that outlined in III.C. Training involving a partially observable network, a network with unknown structure, or one with a very complicated structure would require a significant code rewrite. Once a trained BBN is available, a modified version of TinyDB can be used to perform similarity metrics on incoming data and post relevant evidence to the BBN, which can be used to provide live inference. A BBN could also be cross-validated against data sets that already exist. The visualization created is useful for social networking, but is quite domain-specific.

V. Conclusion and Future Work

Several paths for future work are evident. First and foremost, it would be interesting to see how well these methods perform on other problem domains. The system can detect human interactions relatively accurately, but there are other problems where Bayesian inference on signals measured for their similarity could be useful. Application to other domains would be a true test of the model's flexibility and versatility.

The development of more advanced inferential tools would be useful. The training algorithms currently support learning in fully observable BBNs with known structure. A framework could be developed to allow use of the EM-eta learning algorithm [40] for partially observable BBNs. Other algorithms could be used to help infer a structure for a BBN rather than requiring the user to select one. Training code could be made more generic; currently, changing the network's structure or the names of the nodes requires changing code so training will work correctly. The use of those algorithms is potentially useful not just in the generic case, but also to the specific problem discussed in this thesis. Ideally, the system would be able to conduct learning without re-compiling code, and with a minimal cognitive load on the user.

The use of more sophisticated models may be fruitful as well. Several efforts in identifying human behavior described in the Background section make use of Hidden Markov Models. The problem of identifying face-to-face interaction can be viewed as an HMM. The Boolean interaction state of two individuals can be viewed as the hidden state, and the sensor values that result are the emitted, observable state.

The current software structure allows a user to "plug in" different filters, transforms, similarity metrics, and BBN structures. Further experimentation with those parameters can lead to more accurate results, and the framework for doing so allows rapid analysis of a method's accuracy.

This method of monitoring face-to-face interactions with sensor networks represents a new way to construct social networks, but the question of whether it is possible to classify an interaction more specifically still remains. As it has been developed, the model outputs a certainty rating that is representative of whether or not two specific people are interacting. With proper refinement and training, it may be possible to characterize the interactions further. Data features may indicate that a lecture is going on, as opposed to a round-table discussion. Frequency and duration of the speech patterns of certain individuals may provide clues about asymmetric relations between them. The model currently does not have a very broad view of time, per se: it manages a time window of the last ten seconds, and considers only the very last guess it made for a given pair of individuals. Certainly, a system capable of more sophisticated inference would have a much more intelligent internal timeline, keeping track of past events.

Another problem that needs to be mitigated is training's dependence on its environment. As discussed before, the effectiveness of inference decreases if the system is trained on an environment that differs greatly from the one in which it performs inference. It is possible to structure the system to adjust to a new environment, keeping track of the median values for each attribute that it has seen so far in its existence. Such a system would alter bin values over time based on what it has seen. However, the added flexibility of such a system comes with a consistency tradeoff. If the sensors are exposed to a certain environment for a long time, then the environment switches, it will take time for the system to accustom itself to the new surroundings. It may be better to keep static bin values that are trained, and possibly edited by users. Certainly, this technique merits investigation, and different situations will probably require different binning techniques.

In conclusion, this thesis outlines the development of methods that combine Bayesian inference with signal processing techniques to make inferences about real-world phenomena from sensor networks' data traces. Performance was validated through its application to the domain of social network identification. Berkeley motes were attached to individuals as they performed scripted interactions, and the system was trained on data traces that were collected. The resulting model was able to identify intervals of human interaction reasonably accurately. This work paves the way for more novel methods of social network construction and analysis, and gives hope that these methods may be useful in other ways.

References

1. Crossbow, Inc. Wireless sensor networks (mica motes). http:// www. xbow.com/Products/Wireless Sensor Networks.htm.

2. S. Madden. *The Design and Evaluation of a Query Processing Architecture for Sensor Networks.* PhD thesis, UC Berkeley, 2003.

3. R. Kling, R. Adler, V. Hummel, and J. Huang. Intel Mote Project web page. http://www.intel.com/research/exploratory/motes.htm

4. K. M. Carley, *On the Evolution of Social and Organizational Networks.* In S. B. Andrews & D. Knoke (Eds.), Vol. 16 Special Issue of Research in the Sociology of Organizations. on "Networks In and Around Organizations." (pp. 3-30), 1999. Stamford, CT: JAI Press, Inc.

5. TinyOS web page. http://www.tinyos.net

6. D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brwer, and D. Culler. T*he nesC Language: A Holistic Approach to Networked Embedded Systems.* In Proceedings of Programming Language Design and Implementation (PLDI) 2003, June 2003.

7. S. Gourley. *Soldier Armed: Unattended Ground Sensors.* In Army, Aug 2004.

8. A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson. *Wireless Sensor Networks for Habitat Monitoring.* First ACM Workshop on Wireless Sensor Networks and Applications. September 2002.

9. J. Kumagai. *The Secret Life of Birds.* IEEE Spectrum, April 2004.

10. G. Huang. *Casting the Wireless Sensor Net.* In Technology Review, July 2003.

11. S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications.* Cambridge University Press, 1994.

12. J. Moreno. *Who Shall Survive?,* Beacon, NY: Beacon House Inc., 1953.

13. S. Milgram. *The small-world problem.* Psychology Today 1, 61-67, 1967.

14. J. Pfautz, E. Roth, K. Jones, E. Hudlicka, T. Fichtl, B. Karabaich, G. Zacharias. *Design and Evaluation of a Visualization Aid for Stability and Support Operations.* Found in D. Hancock, M. Mouloua, and P. Vincenzi, eds., *Human Performance, Situation Awareness, and Automation: Current Research and Trends, Volume II.* Lawrence Erlbaum Associates, Mahwah, NJ, 2004.

15. A. Degenne and M. Forse, *Introducing Social Networks.* London: Sage Publications, 1999.

16. InFlow home page. http://www.orgnet.com/

17. Spoke home page. http://www.spoke.com/overview/

18. Visible Path home page. http://www.visiblepath.com/

19. J. R. Tyler, D. M. Wilkinson, B.A. Huberman, *Email as Spectroscopy: Automated Discovery of Community Structure within Organizations.* HP Laboratories, 2003.

20. The Facebook home page. http://www.thefacebook.com

21. Myspace home page. http://www.myspace.com

22. Friendster home page. http://www.friendster.com

23. Orkut home page. http://www.orkut.com

24. LinkedIn home page. https://www.linkedin.com/

25. J. Lowrance. LAW: Link Analysis Workbench home page. http://www.ai.sri.com/~law/index.html

26. L. Freeman. *Uncovering Organizational Hierarchies.* Computational and Organizational Theory 3(1): 5-18.

27. N. Friedkin. *Norm Formation in Social Influence Networks.* Social Networks 23:167-189.

28. D. Jensen and J. Neville. *Data Mining in Social Networks.* National Academy of Sciences Symposium on Dynamic Social Network Analysis, 2002.

29. The Internet Movie Database. http://www.imdb.com

30. R. Hauck, M. Chau and H. Chen. *COPLINK: Arming Law Enforcement with New Knowledge Management Technologies.* In W. McIver and A. Elmagarmid (Eds.), *Advances in Digital Government: Technology, Human Factors, and Policy*, Kluwer Academic Publishers, April 2003.

31. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2003.

32. J. Halpern. *Reasoning About Uncertainty.* MIT Press, Cambridge, MA, 2003.

33. S. Parsons. *Quantitative Methods for Reasoning about Uncertainty.* MIT Press, Cambridge, MA, 2001.

34. J. McCarthy and P.J. Hayes. *Some Philosophical Problems from the Standpoint of Artificial Intelligence.* From B. Meltzer, D. Mitchie, and M. Swann, eds., *Machine Intelligence 4.* Edinburgh University Press, Edinburgh, Scotland, 1969.

35. F. Lin. *An Argument-Based Approach to Non-Monotonic Reasoning*. Computational Intelligence 9, 254-267, 1993.

36. L. Zadeh. *Fuzzy Sets.* Information and Control 8, 338-353, 1965.

37. J. Pearl and S. Russell. *Bayesian Networks.* 2000. Available at: http://www.kddresearch.org/Resources/Papers/Intro/pearl-bbn2000.pdf

38. J. Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, 2001.

39. W. Brunette, J. Lester, A. Rea, G. Borriello, *Some sensor network elements for ubiquitous computing*. IPSN 2005.

40. K. Murphy, *Dynamic Bayesian Networks: Representation, Inference and learning.* PhD thesis, UC Berkeley, Computer Science Division, 2002.

41. M. Brand, N. Oliver, A. Pentland, *Coupled Hidden Markov Models for Complex Action Recognition*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, p. 994, 1997.

42. L. Bao and S. S. Intille, *Activity recognition from user-annotated acceleration data,* in *Proceedings of PERVASIVE 2004,* vol. LNCS 3001, A. Ferscha and F. Mattern, Eds. Berlin Heidelberg: Springer-Verlag, 2004, pp. 1-17.

43. P. Lukowicz, J. A. Ward, H. Junker, M. Stager, G. Troster, A. Atrash, T. Starner, *Recognizing Workshop Activity Using Body Worn Microphones and Accelerometers.* In A. Ferscha and F. Mattern (Eds.): PERVASIVE 2004, LNCS 3001, pp. 18-32, 2004.

44. A. Oppenheim, A. Willsky, S. Hamid Nawab. *Signals and Systems.* Prentice Hall, 1996.

45. P. Buonadonna, D. Gay, J. Hellerstein, W. Hong, and S. Madden. *TASK: Sensor Network in a Box*. European Workshop on Sensor Networks, 2005.

46. PostgreSQL home page. http://www.postgresql.org.

47. BNet.Builder homepage. http://www.cra.com/commercial-products-services/bnet-builder.asp

48. K. Devlin. *Logic and Information.* Cambridge University Press, New York, NY, 1991.

49. E. Castillo, J. M. Gutierrez, A. S. Hadi, *Sensitivity Analysis in Discrete Bayesian Networks.* IEEE Transactions on Systems, Man, and Cybernetics, Vol. 27, No. 4, July 1997.