Illuminac: Simultaneous Naming and Configuration for Workspace Lighting Control

Author 1 Affiliation
Affiliation
author@a.com

Author 2 Affiliation
Affiliation
author2@b.com

ABSTRACT
This paper explores "natural speech" interfaces in an ubiquitous computing environment, specifically, the invocation of custom lighting patterns in an "open-plan" workplace. The workspace contains a large array of individually-dimmable lights which is very flexible but expensive to configure for common tasks. We argue speech is a good solution for this task, but there is a challenge in supporting natural interaction which is shared with other ubicomp environments: discovering the lighting scene names, light configurations and, mappings between them. We describe the Simultaneous Naming And Configuration (SNAC) approach to address this problem and demonstrate its applicability to our system, Illuminac.

Author Keywords
Natural Speech Interfaces, Non-negative Matrix Factorization

ACM Classification Keywords
H.5.2 User Interfaces:{Voice I/O, Natural language}

INTRODUCTION
The number of electronic devices that control our environment is ever increasing. While this trend brings greater flexibility and control over our environment, configuring each individual device to achieve the desired environmental state becomes ever more tedious and often burdensome. For example, in the home, to prepare the environmental state for cooking, one might want to turn the radio on and turn it to the news, turn the volume of the speakers up since the kitchen will be noisy, and make the kitchen lights bright. Then, to prepare the environmental state for eating dinner, one might dim the lights in the dining room, turn on the mp3 player to play a dinner music playlist, turn the volume of the speakers down, and close the blinds. Controlling all of these devices—the lights, the radio, the speaker volume, the mp3 player, and the blinds—to achieve a desired environmental state is quite tedious.

As with any interface, an interface for controlling the environmental state in the home should match the user's mental model instead of that of the underlying devices. That is, the user should only need to specify the *name* of the environmental state rather than the *configuration* of each individual device needed to achieve the desired state. In the home example, the user should be able to say "set cooking mode" or "apply dinner mode" rather than specifying the configuration of the radio, speakers and lights to achieve the cooking or dinner environmental state.

The challenge in designing such an interface that matches the user's mental model is not only in discovering the *names* for the environmental states, but also discovering the configurations of devices necessary to achieve a named environmental state.

In other words, we are only given the set of devices that can be controlled and how to control them, but we do not know the names of the desired environmental states, the configuration of devices to achieve each environmental state, nor the mapping from environment names to device configurations. We call this issue the name-configuration mapping problem. To provide an ideal natural interface, the users should be able to customize the environment names and the corresponding configurations of devices. Therefore, the name-configuration mapping problem cannot be solved *a priori*, but must be solved for each individual set of users and domains.

In addition to matching the user's mental model, we want the interface to be "calm." That is, the interface in a ubicomp environment, like environment control, should be almost invisible except during direct (focal) interaction (as advocated by Weiser [22]). In this context, a speech-based interface seems like a good option [1]. With distributed microphone technology, the physical interface all but disappears, but jumps fluidly to the foreground when the system responds to spoken input. Furthermore, speech is often considered the most natural form of human expression, and has the potential to address certain accessibility concerns.

[1]Weiser explicitly critiqued speech-based agent interfaces in one of his well-known papers. However, his specific criticisms related to many aspects of design that we avoid: agent personality, knowledge, an "identity" you interact with and impart human-like traits to. By contrast, our use of speech is highly situated, in a shared context, and in short focal interactions followed by return to invisibility.

Submitted to UBICOMP 2008

Such natural language-based interfaces only exacerbate the name-configuration mapping problem, as it adds a level of uncertainty to the system: either from imprecision in human input or uncertainty in the capturing of human input. For speech interfaces, users have to recall their commands which may result in slight variants. For example, one may say "set mode for cooking" or "please change the environment for cooking" instead of "set cooking mode." The automatic speech recognizer will inevitably have recognition errors, such as recognizing "set cooking mode" as "set cook in low."

We believe this theme occurs in many ubicomp environments. For example, in the workspace the name-configuration mapping problem arises in mapping the "semantic gap" from a configuration command like "presentation" to control the lights, projector, and sound devices. In a workspace with an array of individually controllable lights instead of a bank of lighting all controlled by one light switch, mapping from a configuration command like "Joe's lights on" to turn on the lights over Joe's desk presents the same challenge. This challenge would also be present in mapping a request like "high priority calls only" to a specific subset of the user's contacts who are important enough to trigger a ring response on a mobile device.

In this paper, we look at designing a natural speech interface for workspace lighting control in order to understand and fully address the name-configuration mapping problem. In this case study there are 79 devices (individually controllable lights) and 25 users who have both their own and shared environment names and configurations. We make the following contributions:

- We identify the distinction (semantic gap) between the name of a desired environmental state and the configuration of devices to achieve that environmental state (the name-configuration mapping problem). The name-configuration mapping problem in the presence of uncertainty is the key challenge in designing natural speech interfaces for environment and device control.

- We describe the Simultaneous Naming And Configuration (SNAC) approach to address the name-configuration mapping problem with uncertainty. The key idea is a learning system simultaneously trained on two kinds of data provided directly by the users' names of environments and configurations of devices. (SNAC Approach section)

- We identify an appropriate learning algorithm for simultaneous naming and configuration: non-negative matrix factorization (NMF). (SNAC Approach Learning Model section)

- We show the applicability of our SNAC approach for natural speech interfaces for workspace lighting control by implementing and deploying a SNAC-based system, Illuminac. With Illuminac we see that one or two training points is often sufficient to produce environmental states with mostly correct configurations, thereby providing good accuracy with little training. (Illuminac section)

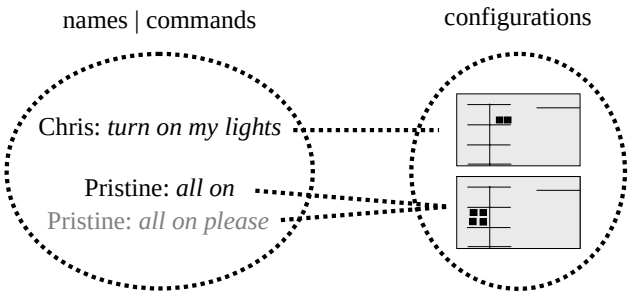

Figure 1. The name-configuration mapping problem for workspace lighting control. The dotted lines indicate the concepts that must be discovered by the system to provide a natural speech-based interface for workspace lighting control.

OVERVIEW OF WORKSPACE LIGHTING CONTROL

Before we present the SNAC approach to the name-configuration mapping problem we first introduce the workspace lighting control domain for which Illuminac is designed.

Many large open plan-workspaces have extensive banks of lights controlled by just one light switch. Therefore, the lighting control is not flexible enough to respond to occupancy or daylighting. Many lights are turned on for just a few occupants, and lights next to a window cannot be turned off without turning off the lights away from the window. More granular control over the lighting in large workspaces could enable a reduction in energy consumption by allowing unnecessary lights to be turned off.

Low-cost granular lighting control systems appropriate for both retrofit and new construction are being developed to enable more flexible lighting control and, as a result, energy savings. One such system developed at the University of California at Berkeley Center for the Built Environment has been licensed to Adura Technologies [21] and is now commercially available. Given this type of technology, more flexible lighting control is possible in many workspaces with large arrays of light fixtures.

Intelligent systems for controlling workspace lights that adapt to daylight from windows and occupancy levels are being developed using flexible lighting control. At the same time, there is still a need for user interfaces that allow users to directly control flexible workspace lights and override such intelligent systems when appropriate.

We propose a speech-based interface for lighting control in shared workspaces, as they are natural and allow for a calm interface. To do so, users should be able to customize the system to work with names of lighting scenes that are natural to them. Based on our experience with Illuminac, we found that one user says "turn on my lights" to turn on the two lights over her desk, while another user says "all on" to turn on the four lights around her desk.

As alluded to earlier, to support customized speech commands for personalized lighting configuration, we must address the name-configuration mapping problem in the pres-

ence of uncertainty. Specifically, we need to discover the names of lighting scenes, the configurations of lights needed to create the lighting scenes, and the mapping between the names and configurations. In Figure 1, we depict the name-configuration mapping problem in the context of workspace lighting control. On the left, we show names of lighting scenes (i.e., user-defined commands for changing the lighting scene), while on the right, we show lighting scene configurations. A lighting scene configuration is depicted as a schematic view of our workspace setup with 6 cubicles where the black squares show the lights that are turned on. The lines show the desired mapping from commands to configurations. For example, we need to discover that Chris says "turn on my lights" to instruct the system to turn on the two lights above his desk, and Pristine says "all on" to turn on the four lights above her desk. More precisely, the system must be made aware of the following:

1. The commands "turn on my lights" and "all on";
2. The configuration with two lights in the top-right cubicle and the configuration with four lights in the middle-left; and
3. That the command "turn on my lights" refers to the configuration with two lights in the top-left cubicle and "all on" refers to the configuration with four lights on in the middle-left cubicle.

Moreover, we would like the system to perform a reasonable scene change (i.e., be robust) even with slight variants of the expected commands. For example, if Pristine says "all on please" instead of "all on", the system should still turn on the four lights above her desk. This variant is shown in Figure 1 as the gray command.

The Simultaneous Naming And Configuration (SNAC) approach that we propose is targeted to address these challenges. To be robust in the presence of human imprecision or imprecision in the capturing of human input, we use a learning-based system. To address the name-configuration mapping problem, we train the system on two kinds of data from the users simultaneously: the commands and the configurations. While the details of the design and evaluation of Illuminac are discussed in the subsequent sections, in the remainder of this section we sketch how the user interacts with our system.

To add a command to Illuminac, users train the system by first recording their command. Then, the user demonstrates the desired lighting configuration and identifies herself. The novel aspect of our system is that rather than simply storing this mapping from command to configuration, we combine the recorded speech command and lighting configuration into a common representation to provide as input for a standard machine learning algorithm. Intuitively, the system uses the learning algorithm to identify structure across the space of command-configuration pairs, not just the space of commands. Once the user has trained the system on a few examples, the user can say her command into any of the microphones in the room, and the system changes the lighting scene by applying the trained model to the user's command. Because the model is trained on commands and configurations specific to the workspace, we expect to be able to perform reasonable lighting actions with less command training. For example, when a visitor who has never provided training input to the system comes into the lab, she can try her command and potentially get reasonable behavior because regular users may have already trained the system on similar commands. Of course, if the resulting behavior is undesired, she can manually change the lighting scene, thereby giving the system another training data point suited to her.

SNAC APPROACH

As we described above, the challenge in supporting customized commands for configuration tasks is not only discovering how users give commands. The challenge is in simultaneously discovering (i) the commands natural to the users, (ii) the configurations naturally used in the domain and (iii) the mapping between the commands and configurations.

Traditional Approach to Designing Speech Interfaces

Discovering the names naturally used by users is a common problem in the design of speech interfaces. Speech interface designers commonly use the wizard-of-Oz data collection technique [6] to gather formative data including sample utterances which helps inform the design of the speech interface toward supporting more natural speech input. The wizard-of-Oz data collection technique allows more realistic sample usage data to be collected before a prototype of the system is ready. A human (called the wizard) acts as the speech interface, responding to the user's commands. The wizard often communicates with the user through a speech synthesizer to make the user think she is using a working system and invoke more realistic responses.

Simultaneously Discovering Naming and Configuration

For workspace lighting control, we not only need to discover the names users use to refer to lighting scenes, but also the configurations of lights that achieve the named lighting scene. Thus, in addition to collecting sample utterances from users, we also collect sample configurations.

Since the configurations are not known *a priori*, a wizard is not able to accurately respond to a user's commands. In fact, if a wizard is used, the users will alter their commands so that the wizard will understand the configurations they are referring to. We discovered this effect during an early phase of data collection in which users sent messages to a wizard asking her to change the lighting scene. Thus, in the SNAC approach, users train the system directly by demonstrating their configurations in addition to recording their commands.

In order to allow for some speech recognition errors, or to be able to make a reasonable guess at a command from a guest to the space, we need an approach that is more forgiving than simply programming a fixed mapping between commands and configurations. In the SNAC approach, a supervised learning algorithm is trained on data provided by

the users. The training data includes both the users' customized commands and their customized configurations.

This approach also allows an interface to support more natural interaction through customized commands and configurations in a new workspace or with a new set of users without having to have a designer do the customization.

SNAC APPROACH LEARNING MODEL

We select a learning algorithm that allows factors to overlap. This is important in our problem because both configurations and commands can overlap. In the home example, a cooking environmental state and a breakfast environmental state could both set the radio to the news station. For workspace lighting, the space is shared and users who sit next to each other often have overlapping sets of lights in their lighting scenes. The names of the configurations can also overlap. In the workspace lighting control example, many users refer to their lights as "my lights." We disambiguate such similar names with the speaker's identification.

NMF finds non-orthogonal or possibly overlapping, factors in the training data. Each data point (a name / configurations pair) can be explained by an additive combination of the factors. NMF is more appropriate than a clustering learning algorithm such as k-means which would assign a light to one cluster, not allowing it to be part of multiple clusters.

Since we need to learn the commands, configurations, and the mapping, we train our model on command-configuration pairs. That is, we train the learning model on two kinds of data simultaneously: the commands and the light configurations. To record a new training point, users provide both the command and the configuration — they:

1. record their command using a microphone
2. demonstrate the lighting scene named by their command by manually changing the lighting scene (using a web-based graphical user interface).

By collecting both kinds of data simultaneously, we are collecting data about the commands that are natural to users, the configurations that naturally occur in the target domain, and the ground truth mapping between the two.

ILLUMINAC

We designed and deployed a SNAC-based natural speech interface for workspace lighting control, Illuminac, to show the applicability of our SNAC approach for workspace lighting control. In this section we describe the workspace for which Illuminac was designed, outline the iterative design steps used in the design process, describe the design of the system, describe the evaluation of the system, and present the results from the evaluation.

Workspace details

Illuminac was designed for and is deployed in, a 2300 square foot open plan shared workspace with about twenty-five regular occupants (nineteen of whom have permanent desks in the workspace, the rest have permanent desks in the adjacent room). The workspace has six graded-awareness cubicles (cubicles with walls of varying heights from full height to desk height) that occupy half the room. The other half is a multi-use space for meetings, presentations, ad-hoc team meetings or individual work.

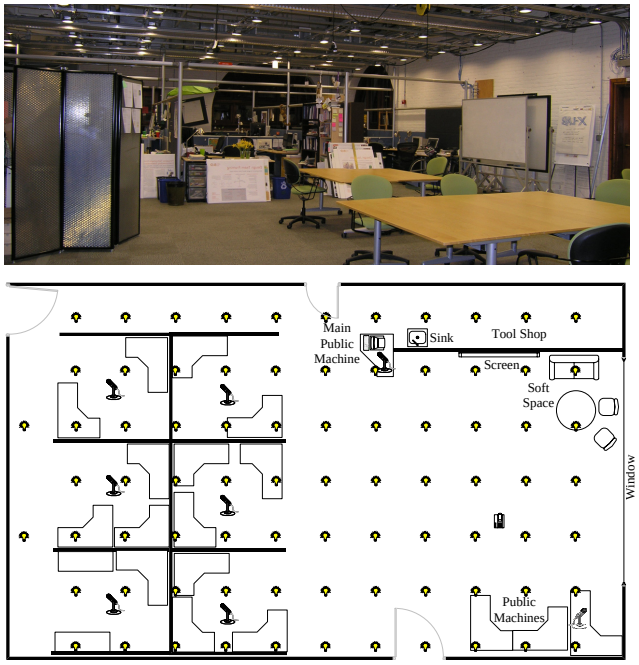

Figure 2. Picture and floor plan of the workspace for which we have implemented a natural speech interface for the lighting control. Each of the 79 individually-controllable lights as well as the eight microphones are shown.

The multi-use space has a presentation screen, a "soft space" with a couch and chairs, and a set of four computers for visitors to the lab to use. There is also a tool shop in one corner of the room. Figure 2 shows a picture and the floor plan of the room. The room has 79 individually-controllable compact fluorescent lights mounted overhead. The intensity of each of the lights can be controlled over the network via a web interface. All occupants of the lab have access to the web interface (Figure 3). They can access it from their personal computers, or from one of the public machines.

The public machine next to the entrance (labeled"Main public machine" in Figure 2) always has the web interface open. There are eight desk microphones throughout the room to allow easier access to the lighting control system. There is one microphone in each cubicle, one at the desk in the corner, and one at the main public machine where the graphical interface to the lights is always open in a browser window. Each microphone has a clearly labeled on/off switch so that residents may control what is and is not recorded.

Iterative Design Steps

In the design of our natural speech lighting control system we followed an iterative design process beginning with a text based wizard-of-Oz study, followed by a training data

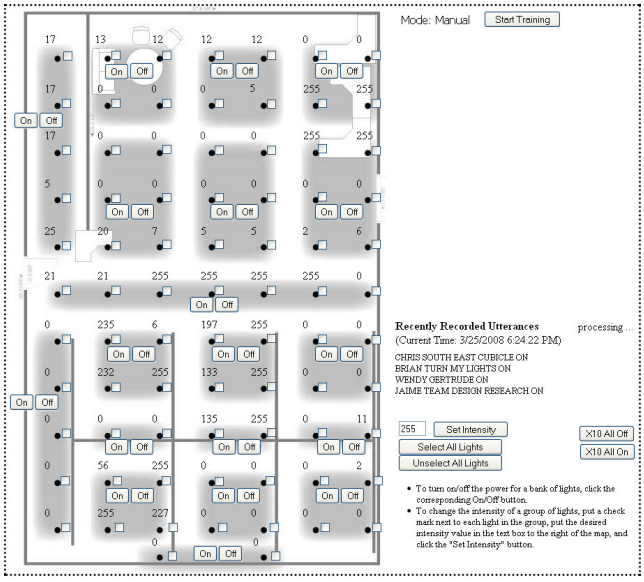

Figure 3. Web-based graphical user interface used to manually configure the array of lights in the workspace.

collection study and finally a deployment of the live system.

Text Based Wizard-of-Oz Study

We began our design of Illuminac with a low-fidelity wizard-of-Oz study to better understand the lighting control domain including the kinds of lighting scenes used in the space and the types of commands used to refer to the lighting scenes. By low-fidelity we mean natural language text input instead of speech input. Participants were asked to send the wizard (a researcher in the lab) an instant message whenever they wanted to change the lighting scene. The wizard would change the lighting scene using the web interface based on the participant's message. If the wizard was not available to change the lighting scene, the participants were asked to type the message they would have sent to the wizard into the web interface and then change the lighting scene themselves with the web interface. This way data was still able to be collected if the wizard was not at his desk. The GUI interface was similar to the one in Figure 3, but it included a text box for entering the messages.

We collected three weeks of data including 230 command / configuration pairs from ten participants who were regular occupants of the space.

This study confirmed our hypothesis that the lighting configurations are sometimes overlapping and are not all disjoint sets of lights. After the formal study concluded, some participants expressed the desire to continue being able to ask the wizard to change the lighting scene. But instead of sending instant messages they wanted to ask the wizard with a spoken command. This provided anecdotal evidence that speech would be a good fit for lighting control in the space.

Formative Training Data Collection

After the low-fidelity wizard-of-Oz study we collected two weeks of high fidelity training data to design and tune the learning algorithm for estimating lighting scenes given a spoken command. The study included sixteen participants who were regular occupants of the space. Each participant was asked to record a command and demonstrate the desired system response as if they were training the system to understand their personalized commands. They were asked to complete the following three steps each time they wanted to change the lighting scene:

1. Say their command to change the lighting scene
2. Type their name into the text box on the web page
3. Change the lighting scene with the web interface

We did not tell them when or how to change the lighting scene in the lab. The participants could use any of the eight microphones throughout the space to record their command. Then they used the web interface to demonstrate their desired change in the lighting scene. We followed the data collection with individual interviews, and asked the participants to reflect on their lighting control preferences and their experience with the study.

We collected 120 command/configuration pairs. For each pair we collected the user's name, the audio clip of the command and the intensity values before and after the command. Each audio clip was transcribed manually as well as with an automatic speech recognizer. The complete vocabulary included about 350 unigrams and bigrams.

This high-fidelity formative training data once again demonstrated both the configurations and commands are overlapping, validating the selection of NMF as the basis for the learning algorithm. This data directly informed the design of the learning algorithm which in turn enabled us to build and deploy the live system.

System Design

Illuminac has two modes, a training mode where users train the system on their personalized lighting configurations and a running mode where users can use their personalized commands to change the lighting scene. The system is in running mode most of the time, ready to process a command and change the lighting scene. To switch to the training mode, the user uses the "Start Training" button on the web-based graphical user interface (shown in Figure 3).

Training Mode

In the training mode, users complete the following four steps to add a training point. Figure 4 shows the four steps.

1. Record her command using one of the microphones around the room.
2. Correct any recognition errors in the automatic speech recognition transcript using the web-based graphical user interface.
3. Demonstrate her desired lighting scene using the web-based graphical user interface to manually set the lighting configuration.

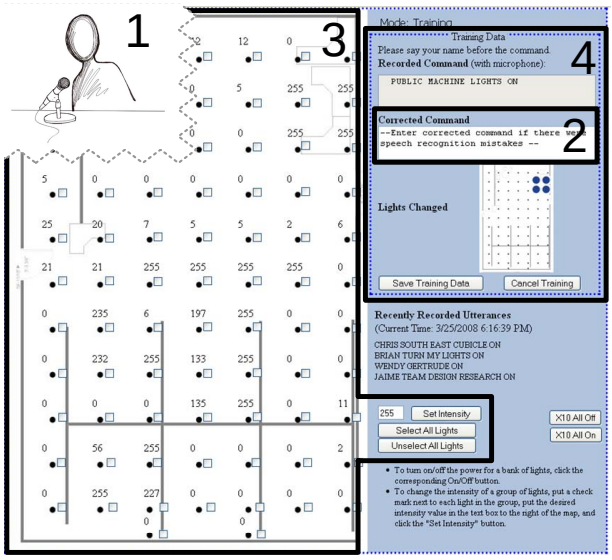

Figure 4. The four steps users follow in the training mode to train the system on their personalized command and lighting scene.

4. Verify the command and configuration are correct before saving the training point.

After the user adds a new training point, the system retrains the model on the old training data plus the new data point. It also adds the command to the speech recognition language model to increase the recognition accuracy for the command the next time it is used. The new model is trained and ready to be used in the running mode in just a few seconds. Even though we are not using an online training algorithm, we can retrain the model fast enough when we get a new training point that users can think of the system as an online training system.

Running Mode

In running mode, a user can use one of her speech commands to change the lighting scene. To do so the user says her command into any one of the microphones around the room. The system transcribes the command, applies the trained model to the transcript and changes the lighting scene according to the estimates lighting scene.

Figure 5. An LED next to each microphone lights up when the system is processing an audio command to let users know the system "heard" them.

If the estimated lighting scene is incorrect, the user can say "undo," "cancel," or "wrong" withing three minutes to undo the last lighting scene change.

While the system is transcribing a command, the text "processing ..." is displayed in the GUI and an LED next to each microphone lights up (see Figures 3 and 5). This lets the user know the system "heard" them and is processing the command.

Learning Model

The system uses a learning algorithm, namely least-squares non-negative matrix factorization (NMF) to estimate the new lighting scene given a command.

NMF has been shown to work well for pure text clustering as well as image segmentation [10]. In our problem, we have both text (the commands), as well as an image (the grid of lights).

As the name suggests, NMF imposes a non-negative constraint on the data. The commands are naturally represented with non-negative values in a term-frequency vector. The lighting scene is also naturally represented with non-negative values for the intensity of each light.

The NMF algorithm is fast enough to allow us to retrain the model between uses. The algorithm runs in 0.89 seconds on two weeks of training data.

Given the similarity of our data with text and images, the non-orthogonal factors in our data, and the naturally non-negative values in our data representation, we selected NMF to learn which lights should change given a new command and the current lighting scene.

Data Representation

Commands

We represent the commands (tagged with the user's name) with a term-frequency vector. Each word (unigram) or pair of consecutive words (bigram) that appears in any of the commands is represented with an entry in the vector, thus the length of the vector is the number of unique unigrams or bigrams in all of the commands. The value of each entry in the vector is the number of times the unigram or bigram appears in the command. We include bigrams to capture phrases such as "soft space," "kitchen area," or "public machines."

Lighting Scene

We represent the lighting scene as a vector of intensity values, one for each light. The intensity values that did not change when the user demonstrated the new lighting scene are set to zero to allow the algorithm to learn which lights a command refers to as well as what intensity to set the lights to.

The reader may notice the value zero in the intensity vector could mean a light is not in the set of lights named by the command, or the command is an "off" command which sets the intensity values to zero. Instead of overloading the meaning on zeros, we treat "off" commands differently. When interpreting an "off" command we use the trained model to tell us which lights to turn off. In contrast, when we interpret an "on" command we use the model to tell us which intensity values to change and what values to change them to.

With this special treatment of "off" commands, we can either

leave the off commands out of the training data or "translate" an off command to an on command by using the difference in intensity values instead of the intensity values in the final lighting scene because these values would be zero since it is an off command. The former requires throwing away some of the training data and results in users needing to provide more training points. The latter solution allows us to use all the training data, we use this solution.

Application of NMF

Our trained model is a set of orthogonal, possibly overlapping factors. To get the factors that describe our training data, we use NMF to factorize our data into two matrices, one of which describes the factors in our data.

NMF is an algorithm that finds a positive factorization of the given matrix [11, 10]:

$$\mathbf{X} \approx \mathbf{U}\mathbf{V}^T$$

where $\mathbf{X}$ represents the training data, $\mathbf{V}^T$ represents the factors in our data, and $\mathbf{U}$ tells the strength of each factor in each of the data points.

Each row in $\mathbf{X}$ is a training point (command / configuration pair). Thus each row vector is comprised of the command term frequency vector concatenated with the lighting configuration vector. The matrix $\mathbf{X}$ has dimensions $m \times n$ where m is the number of training points in the model and n is the length of the term-frequency vectors (q) plus the number of lights (p). The matrix $\mathbf{V}^T$ has dimensions $k \times n$ where k is the number of factors. We explain shortly how we chose k in the system. Each row in $\mathbf{V}^T$ represents one of the factors. The matrix $\mathbf{U}$ has dimensions $m \times k$. The i^{th} row in $\mathbf{U}$ gives the coefficients to the additive sum of the factors to describe the i^{th} training point.

Selecting k

A good factorization should explain the data with the smallest number of factors. We used the formative training data to calculate the accuracy rate for a large range of k values and select the smallest k among the values of k with the highest accuracy rates. Based on a plot of the accuracy as a function of k we select k to be 32.

Lighting Configuration Estimation

To estimate a lighting configuration given a new command, we use NMF again to find the factors from our training data that are present in the new command. Figure 6 depicts how to apply the trained model to a new command. As descried above, our trained model is a matrix which represents the factors in the training data, $\mathbf{V}^T$. Each factor has a "command" component, and a "configuration" component. The command components are in the left half of $\mathbf{V}^T$ (columns 1 to q in $\mathbf{V}^T$), the configuration components are in the right half of $\mathbf{V}^T$ (columns $q+1$ to n in $\mathbf{V}^T$). We will call these matrices$\mathbf{V}^T_{cmd}$ and $\mathbf{V}^T_{config}$ respectively.

$$\mathbf{V}^T = \left[\mathbf{V}^T_{cmd}, \mathbf{V}^T_{config}\right]$$

The matrix we need to factorize contains the term-frequency

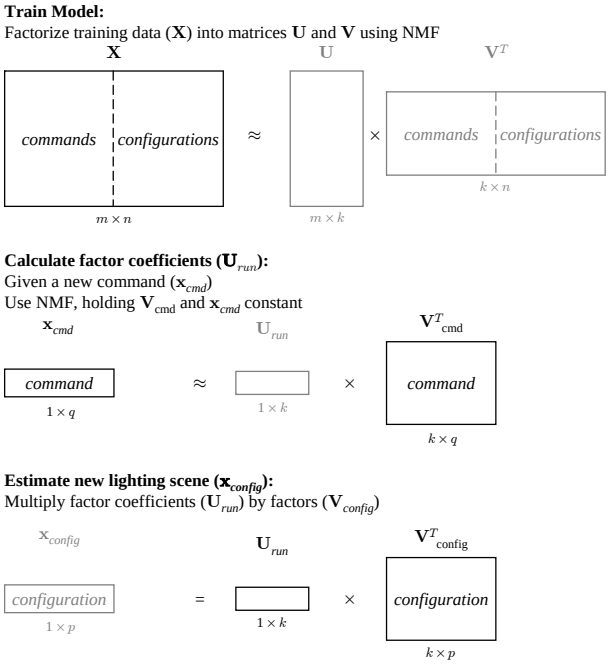

Figure 6. Given a new command, we use the factors matrix from our training data and NMF to estimate the new lighting configuration. In each step, matrices which are given are in black, matrices which we are calculating are in gray.

vector for the new command and is only one row. We run NMF on the command term-frequency vector

$$\mathbf{x}_{cmd} \approx \mathbf{u}_{run} \times \mathbf{V}^T_{cmd}$$

($\mathbf{x}_{cmd}$ has dimensions $1 \times q$, $\mathbf{u}_{run}$ has dimensions $1 \times k$), holding $\mathbf{V}^T_{cmd}$ constant to get $\mathbf{u}_{run}$ which tells us which factors are present in the new command. We can multiply $\mathbf{u}_{run}$ by the contributions component of the factors to get the lighting configuration vector:

$$\mathbf{x}_{config} = \mathbf{u}_{run} \times \mathbf{V}^T_{config}$$

The final step in estimating the new lighting configuration is to estimate which intensity values to change and what value to change them to. We use two NMF models to do so. To estimate which intensity values to change, we train a model on a data matrix where the values in the configuration vectors are boolean values that indicate which lights are part of a lighting configuration. To estimate what intensity value to set a light to, we use the original data matrix described above. When we apply these two models to a new command, we get two parts of the lighting configuration estimate, $\mathbf{x}_{configbool}$ and $\mathbf{x}_{config}$. $\mathbf{x}_{configbool}$ is a vector with values between 0 and 1. We perform cross validation on the formative training data to select a threshold value to convert the values to boolean values. We change the intensity value for each light that has a one in $\mathbf{x}_{configbool}$ to the estimated intensity value for that light in $\mathbf{x}_{config}$. In other words, we only change the intensities of the lights involved in the lighting scene referred to in the new command.

Automatic Speech Recognition

We are using Carnegie Mellon's Sphinx 3 speech recognizer [16] to transcribe the commands. Sphinx 3 is a state-of-the-art fully-continuous acoustic model recognizer which is open for experimental use.

We trained an acoustic model on the ICSI Meeting Corpus [8] which features all the imperfections of natural speech: pauses, um's and ah's, truncated words, grammar errors, sentence and phrase restarts, etc. It also features a variety of nationalities and accents. We feel this acoustic model is a better match for our user base than the publicly available acoustic models (HUB4, WSJ, RM1) which feature artificially clean speech being read from transcripts.

The language model is trained on the commands from the training data. The recognizer runs fast enough to transcribe the commands almost in real-time (it runs in 1.2x real time on a 3 GHz dual processor machine with 1.5 MB of ram). As described above, we use a status LED to let users know the system "heard" them while the recognizer is processing the command.

Evaluation

Study Description

We deployed Iluminac with ten of the twenty five regular occupants of the lab for one week. We started with no training data and asked the participants to train the system on their commands again (many of the participants in this study also participated in previous studies and had already provided training points). Participants were instructed to use the system whenever they wanted to change the lighting scene. The first time they used the system for a particular command they were asked to record a training data point. Subsequent times they were asked to test their commands, recording more training points if the system did not respond as expected. When the participants tested a new command they recorded the accuracy of the results on a paper log next to the microphone. They recorded the accuracy of the system response by circling one of the following options:

4	3	2	1	0
Correct	Partially Correct	Some Correct, Some Wrong	Nothing Happened	Wrong

At the end of the study the participants were asked to complete an anonymous web questionnaire about their experiences with the system.

Data Collected

We collected 43 training points and 81 test points from ten participants. For each training point we collected the state of the lighting scene before and after the training session as well as the command (transcribed using the automatic speech recognizer and corrected by the participant). For each test point we collected the state of the lighting scene before and after the system changed the lighting scene, the command the participant spoke into one of the microphones and the participant's evaluation of the system's response.

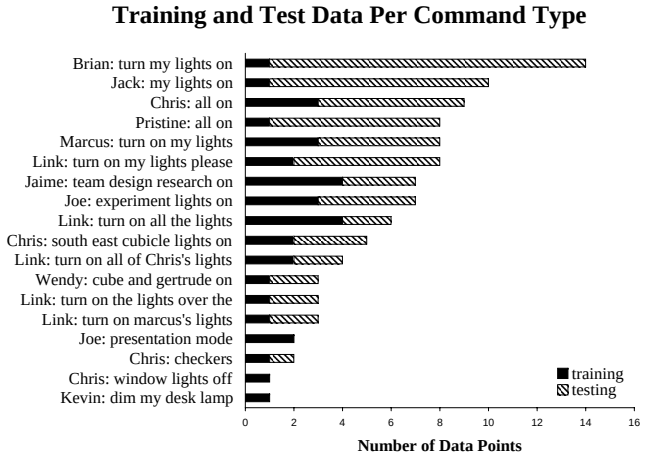

Figure 7. The number of training and test data points collected for each group of similar commands. The command group labels are a representative command recorded by the participants.

Figure 7 shows the number of data points collected for each group of similar commands.

Study Results and Discussion

To analyze the results we manually assigned each training and testing point to a group of similar commands. For example all of the commands Jack used to refer to the lights over his desk were put into one group. Figure 7 lists a representative command for each group of similar commands.

The average testing score plateaued between "correct" and "partially correct" when commands were tested with 1, 2 and 3 training points (see Figure 8). We believe these results could be improved by changing the training GUI to alleviate a common confusion about which lights were being saved as a lighting scene. The interface only recorded the intensity values that changed during the training mode, but users thought it was saving the intensity values for each light selected. As a result some of the training data was not correct, which we believe impacted the accuracy results negatively.

Although the average score is closer to "partially correct" than correct, when asked in the post questionnaire "After the study is over, would you like to continue using the system?" 8 out 10 participants responded *Yes*, and two responded *Maybe* (the options were *Yes*, *Maybe* and *No*). One of the participants who responded *maybe* says the microphone was to far away and he was lazy. Right now each cubicle with three people shares one microphone, this could be remedied by giving each user a microphone at their desk, making the microphone convenient to access. The other participant who responded *maybe* tends to sit in the public area most of the time where the furniture moves around quite a bit and he doesn't often use the same set of lights. In such an open space, a location based speech approach would work much better, where users could say "lights on here." Such a location based approach can be implemented with distributed array technology overhead. Such technology would not be desirable in the cubicle area for privacy reasons. With overhead microphones, users cannot control what is being recorded,

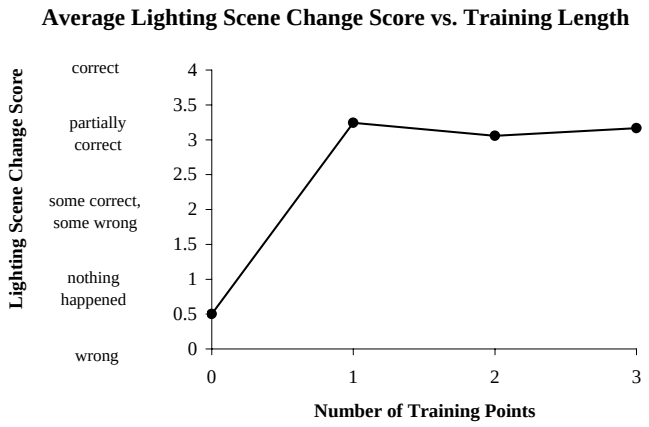

Figure 8. The average test point score as reported by the participants versus the training length for the specific type of command when the test point was recorded

but with desk microphones, users have the power to turn the microphone on their desk off.

When asked "How many training data points would you be willing to provide to be able to use speech to control the lights" participants responded with an average of 3.9 (min 2, max 5). On average participants recorded 1.9 training points per command group during the study. Since the average number of training points participants would be willing to provide is higher than the average number they recorded during a formal study we believe the performance of the system outside a formal study would be similar to the performance during the study.

RELATED WORK

Below we situate our work in the work related to designing natural speech interfaces, existing smart home and home IT projects, multimodal interfaces, and previous applications of factorization algorithms such as NMF.

Natural Speech Interface Design

In speech interfaces where all of the objects that can be referred to are known *a priori*, the designer can use the wizard-of-Oz technique [6] to support names for those objects that are natural to the users. For example, in the RoomLine [3] system, a speech interface which allows users to reserve rooms over the phone, all of the rooms, the sizes of the room, and the equipment available in the rooms is known at the time the speech interface was designed. Such speech interfaces can be grammar-based like many successful commercial systems (BeVocal [15], Tellme [14]), they can be statistical language model - based [4], or they can use a combination of both [18]. In our system, we don't know the objects (configurations) *a priori*, but we still want to be able to support names that are natural to the users, we train a model on both the names for the objects and the objects.

Smart Homes and Home IT

Controlling lighting and other devices in the home is not new, nor is using speech to do so, but we believe using speech to control configurations of devices is novel. Quesada et al. address the interface challenge in the home machine environment [17] with a speech interface for lighting control in the home, but each light is controlled with individual commands, and the speech interface is designed specifically for the particular set of lights. Mozer et. al. [13] have studied predictive light automation, which as mentioned above, could be combined with a speech interface for situations when the predictive light automation does not do a good job with the prediction. Juster and Roy use situated speech and gestures to control a robotic chandelier, Elvis [9]. Their work focuses on how to move the chandelier arms to achieve a desired lighting scene given input from photo sensors. They focus less on the speech interface to the chandelier. It is designed to support a static set of lighting configurations. We focus is on the speech interface, and supporting customized commands and configurations for each new set of devices without having to have a designer do the customization.

Multimodal Interfaces

It is well established that speech and gesture work well together [12] and many of our users mentioned they would like to be able to point at the lights they would like to turn on. Wilson's work on the XWand [23] demonstrates such a system. The user can control different devices by pointing at the device and saying a predetermined utterance. The position of the wand is determined with the use of at least two calibrated cameras and a blinking LED at the end of the wand. The speech recognition system uses a simple command and control style grammar. Wilson acknowledges "while speech clearly has enough expressive power to make the wanted unnecessary, relying on speech alone can be difficult in practice." We aim to address this difficulty. Wilson also acknowledges "the acceptance of the XWand or a related device is limited by the limitations imposed by the installation and calibration of the cameras." The authors address this limitation by trying a wand with audio feedback to aid in pointing tasks without cameras available to track the position of the wand. They find it is possible without the cameras, but the pointing takes more thought on the part of the user, and requires the targets be more widely spaced, which is not the case in our workspace. Our proposed approach could be combined with the XWand to develop a multimodal gesture and speech interface for configuration tasks where the configurations are not able to be predetermined and can also be used in cases where the calibration and installation of cameras is not possible.

Applications of Matrix Factorization

In recent years, alternative factorization methods such as least-squares NMF (Non-negative Matrix Factorization) have found favor over SVD in cases where patterns are not orthogonal. This is especially true when the patterns appear "non-negatively." This is indeed the case for both light intensities and for word frequencies in user commands. So NMF seems like a natural candidate for SNAC analysis. Furthermore, NMF has been shown to be superior to SVD for pure text clustering [10], or for image segmentation [10] which is similar to our light grouping problem. Barnard et al. match words and pictures [1] using an "aspect" probabilistic model,

which is another type of factor model. Other candidate factor models include Latent Dirichlet Analysis (LDA) [2] and GaP [5]. These methods add prior probabilities to the factors and use likelihood measures (rather than least squares) to fit the original data. We did not use these methods because: (i) when there is enough training data, the factor priors have little or no effect and (ii) least-squares NMF has been shown to produce better (more independent) factors compared to KL-divergence (likelihood) fitting methods [10].

CONCLUSION AND FUTURE WORK

This paper introduced an approach to the challenge of simultaneously learning names and the configurations named in natural speech interfaces for environment and device control. We demonstrated this challenge and introduced a solution in the domain of workspace lighting control. Our system allows users to have customized "light switches" for their lighting scene configurations through the use of natural language speech commands. The users train the system not only on their personalized commands but also on the configurations their commands refer to.

Results from the live deployment of our natural speech interface for workspace lighting control are promising and we look forward to seeing how well our approach works in other workspaces, especially ones where the authors don't know all of the occupants. In addition to other workspaces we would like to evaluate our approach with other devices in the workspace in addition to lights such as projectors, speakers, etc, and other ubicomp domains such as home environment control and mobile device configuration.

ACKNOWLEDGMENTS

Omitted for blind review

REFERENCES

1. K. Barnard, P. Duygulu, N. de Freitas, D. Forsyth, D. Blei, and M. I. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.

2. D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

3. D. Bohus. Roomline: a spoken dialog system that provides assistance for conference room reservation and scheduling, as of 19 Sept 2007. http://www.ravenclaw-olympus.org/roomline.html.

4. P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Comput. Linguist.*, 16(2):79–85, June 1990.

5. J. Canny. GAP: a factor model for discrete data. In *SIGIR*, pages 122–129. ACM Press, 2004.

6. N. Dahlbäck and A. Jönsson. Wizard of oz studies – why and how. In *IUI*, pages 193–200, 1993.

7. A. Fiedler and M. Gabsdil. Supporting progressive refinement of wizard-of-oz experiments. In *Proceedings of the ITS 2002 - Workshop on Empirical Methods for Tutorial Dialogue Systems*, pages 62–69, 2002.

8. A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, E. Shriberg, A. Stolcke, A10, C. Wooters, and A11. The icsi meeting corpus. In *ICASSP*, volume 1, pages I–364–I–367 vol.1, 2003.

9. J. Juster and D. Roy. Elvis: situated speech and gesture understanding for a robotic chandelier. In R. Sharma, T. Darrell, M. P. Harper, G. Lazzari, and M. Turk, editors, *ICMI*, pages 90–96. ACM, 2004.

10. D. D. Lee and S. H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 1999.

11. D. D. Lee and S. H. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, volume 13, pages 556–562, 2000.

12. D. Mcneill. *Hand and mind: What gestures reveal about thought*. University of Chicago Press., 1992.

13. M. C. Mozer. Lessons from an adaptive house. In D. Cook and R. Das, editors, *Smart environments: Technologies, protocols, and applications*, pages 273–294. Wiley & Sons, 2005.

14. T. Networks. Fundamentally improving how people and business use the phone, as of 19 Sept 2007. http://www.tellme.com.

15. Nuance. Formerly bevocal, a leading provider of hosted application systems for customer self-service, as of 19 Sept 2007. http://www.bevocal.com.

16. P. Placeway, S. Chen, M. Eskenazi, U. Jain, V. Parikh, B. Raj, M. Ravishankar, R. Rosenfeld, K. Seymore, M. Siegler, R. Stern, and Thayer. The 1996 hub-4 sphinx-3 system. In *In DARPA Speech Recognition Workshop*, Chantilly, VA, February 1997.

17. J. F. Quesada, F. Garcia, E. Sena, J. A. Bernal, and G. Amores. Dialogue management in a home machine environment: Linguistic components over an agent architecture. In *Spanish Society for Natural Language Processing*, volume 27, pages 89–98, September 2001.

18. M. Rayner, P. Bouillon, B. A. Hockey, N. Chatzichrisafis, and M. Starlander. Comparing rule-based and statistical approaches to speech understanding in a limited domain speech translation system. In *TMI*, 2004.

19. M. Rayner, B. A. Hockey, and P. Bouillon. Building linguistically motivated speech recognisers with regulus. In *Tutorial presented ACL*, Barcelona, Spain, 2004.

20. M. Stuttle, J. D. Williams, and S. Young. A framework for dialogue data collection with a simulated asr channel. In *INTERSPEECH*, October 2004.

21. A. Technologies, as of March 2008. http://www.aduratech.com.

22. M. Weiser and J. S. Brown. The coming age of calm technolgy. In *Beyond calculation: the next fifty years*, pages 75–85. Copernicus, New York, NY, USA, 1997.

23. A. Wilson and S. Shafer. Xwand: Ui for intelligent spaces. In *CHI*, pages 545–552, New York, NY, USA, 2003. ACM Press.