Illuminac: Simultaneous Naming and Configuration for Workspace Lighting Control

Ana Ramírez Chang, John Canny
Berkeley Institute of Design and Computer Science Division
University of California, Berkeley
{anar, jfc}@cs.berkeley.edu

ABSTRACT

We explore natural and calm interfaces for configuring ubiquitous computing environments. A natural interface should enable the user to *name* a desired *configuration* and have the system enact that configuration. Users should be able to use familiar names for configurations without learning, which implies the mapping from names to configurations is many-to-one. Instead of users learning the environment's command language, the system simultaneously learns common configurations and infers the keywords that are most salient to them. We call this the SNAC problem (Simultaneous Naming and Configuration). As a case study, we design a speech interface for workspace lighting control on a large array of individually-controllable lights. We present an approach to the SNAC problem and demonstrate its applicability through an evaluation of our system, Illuminac.

Author Keywords
Natural Speech Interfaces, Non-negative Matrix Factorization, Environment Control

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

INTRODUCTION

The number of electronic devices in our environment is ever increasing. While this brings greater flexibility and control, configuring each individual device becomes ever more tedious. 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 watching television, one might dim the lights in the living room, turn on the television, turn the volume of the speakers to medium, and close the blinds. Controlling all of these devices—the lights, the radio, the television, the speaker volume, the television, and the blinds—to achieve a desired environmental state is quite tedious. It is widespread best practice to use activity-specific "configurations" of many devices rather than setting each device individually. The user can then invoke the configuration with a single action— i.e. a key press or a spoken command.

As with any interface, an interface for controlling the environmental state in the home should match the user's mental model. That is, the user should only need to specify an intuitive *name* for 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 indicate "set cooking mode" or "apply TV settings" rather than specifying the configuration of the radio, speakers and lights to achieve the cooking or television watching environmental state.

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 [12]). In this context, a speech-based interface seems like a good option. 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.

In the home example, the user should be able to say "I'd like the cooking mode please", or "prepare the room to cook dinner," or similar variation, and have the system give a similar response. They should also be able to say "cooking mode" or "I'd like to watch TV" and get a different response. These terms are widely shared by people, and their repeated use during training allows a system to learn them as well.

Note that this problem is more challenging than simply memorizing command strings and the appropriate device settings. In the latter case, the system will be extremely brittle, and will respond only when exact training strings are provided. By *simultaneously* learning commands and device settings, the system becomes both more robust and better able to generalize. We called this the SNAC problem (Simultaneous Naming And Configuration).

We believe this theme occurs in many ubicomp environments. For example, in the workspace the SNAC 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 “Turn Joe’s desk lights on please” to turn on the lights over Joe’s desk presents the same challenge.

In this paper, we describe the design, deployment and testing of a natural speech interface in an open-plan workspace. The system runs live and controls 79 devices (individually controllable lights) from 25 users who have both their own and shared environment names and configurations. We make the following contributions:

- We identify and describe an approach to the Simultaneous Naming and Configuration problem (SNAC) — a problem which arises in natural environment control.
- We identify an appropriate learning algorithm for the simultaneous naming and configuration problem: non-negative matrix factorization (NMF)
- We show the applicability of our approach to the SNAC problem on 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).

OVERVIEW OF WORKSPACE LIGHTING CONTROL

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 far 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. In addition to turning off unnecessary lights, Moore *et al.* show when users are given control over their lights, they often set the lights to a luminance level lower than the level recommended by the Chartered Institution of Building Services Engineers (CIBSE) with an average lamp output 55% of the maximum [8].

Proprietary granular lighting control systems for both retrofit and new construction have been available for some years [1]. Most recently, a low-cost industry standard has emerged: DALI: Digital Addressable Lighting Interface – is a bus system for individual light control which has been adopted by most lighting manufacturers, and deployed in major installations (e.g. Heathrow terminal 5). It is an increasingly popular option in energy-conscious building design. However, the increase in flexibility implies an increase in control complexity. The current state-of-the-art in complex lighting control is panels of wall buttons with (often cryptic) scene names, or touch screens with complex menu hierarchies. Neither of these approaches address the issue of lighting in reconfigurable workspaces, where many more configurations are possible. There is clearly a need for more flexible, intuitive and scalable lighting control.

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. Escuyer *et al.* found users didn’t mind the occasional error in an automatic lighting control system as long as they had an easy manual way to correct the lighting scene [4]. Love notes that automatic controls were commonly disabled by users [7]. Whether it’s in conjunction with an intelligent system, or on it’s own, a manual control for flexible lighting is necessary.

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. For example, 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 “team design research lights on” to turn on the four lights around her desk. In our approach to the SNAC problem we use a learning-based system that is trained on two kinds of data from the users simultaneously: the commands and the configurations (lighting scenes). 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 and lighting scene to Illuminac, users train the system by first recording their command. Then, the user demonstrates the desired lighting configuration and identifies herself. In our experience the user only needs to provide two or three training points to get good results. 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, it identifies 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 add a training point.

APPROACH TO THE SNAC PROBLEM

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, and (ii) the configurations named by those commands.

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 [3] to gather formative data including sample utterances, which helps inform the design of the speech interface toward supporting more natural speech input.

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, we collect sample configurations for those utterances.

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 to which they are referring. 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 our approach, users train the system directly by demonstrating their configurations in addition to recording their commands.

To make the system both more robust to small variances in the commands and better able to generalize, we use a supervised learning algorithm. The learning algorithm is trained on data provided by the users that includes both the users' personalized commands and their customized configurations. This approach allows an interface to support natural interaction through customized commands and configurations in a new workspace or with a new set of users without having to have a designer perform the customization.

LEARNING MODEL

We select a learning algorithm that allows factors to overlap. This property is important in our problem because both configurations and commands can overlap. In the home example, a cooking environmental state and a television watching environmental state could both set the speaker volume up. 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.

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

ILLUMINAC

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

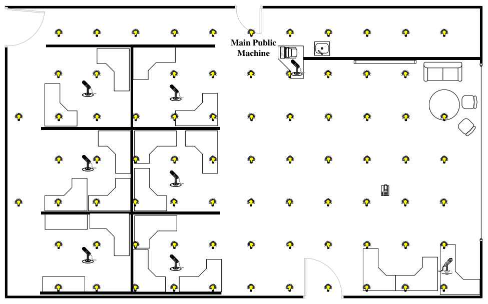

Figure 1. Workspace for which Illuminac was designed.

Workspace Details

Illuminac was designed for and is deployed in a 2,300 square foot open-plan shared workspace (shown in figure 1) with about twenty regular occupants. The workspace has six graded-awareness cubicles that occupy half the room. The other half is a multi-use space for meetings, presentations, ad-hoc team meetings or individual work. The room has 79 individually-controllable compact fluorescent lights mounted overhead and eight desk microphones throughout the room.

Iterative Design Steps

In the design of our system, we followed an iterative design process beginning with a three week lo-fidelity text-based data collection study, followed by a two week hi-fidelity training data collection study, and finally a deployment of the live system. To insure we got data about the commands and configurations that occur naturally in the space we used an *in situ* study design with regular occupants of the space as participants. See [2] for more details.

Learning Model

Data Representation: Commands. We represent the commands (tagged with the user's name) with a term-frequency vector. A term is a word (unigram) or pair of consecutive words (bigram) that appears in any of the commands, and each term 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 be able to capture phrases, such as "soft space," "kitchen area," or "public machines."

Data Representation: 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 to which lights a command refers, as well as to what intensity to set the lights.

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 [6]: $\mathbf{X} \approx \mathbf{U}\mathbf{V}^T$, 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 (i.e., a 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. 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 the Number of Factors (k). A good factorization should explain the data with the smallest number of factors. We used the formative training data to calculate the accuracy for a large range of k values (between 1 and 200) and plotted the accuracy as a function of k. We chose the value of k where increasing k had negligible improvement in accuracy (i.e., the elbow of the curve), which we found 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. As described above, our trained model is a matrix that 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 $\mathbf{V}^T$), and 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 where $\mathbf{V}^T = \left[\mathbf{V}^T_{cmd}, \mathbf{V}^T_{config}\right]$.

The matrix we need to factorize contains the term-frequency vector for the new command, which 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 then multiply $\mathbf{u}'_{run}$ by the configurations 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 to what value to change them. 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 to what intensity value to set a light, 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 $\mathbf{x}'_{config}$. The $\mathbf{x}'_{configbool}$ vector has 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 3.5 speech recognizer [10] to transcribe the commands. We trained an acoustic model on the ICSI Meeting Corpus [5], which features all the imperfections of natural speech: pauses, um's and ah's, truncated words, grammar errors, sentence and phrase restarts, etc. The language model is trained on the commands from the training data.

Evaluation

We deployed Iluminac with ten of the twenty five regular occupants of the lab for one week. The system ran with live speech recognition on the audio from the microphones around the room, live training mode where the model was retrained after each training point was collected, and live running mode, which applied a lighting scene when a user spoke a command into one of the microphones. We started with no training data and asked the participants to train the system on their commands again. 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, but 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 - Correct; 3 - Partially Correct; 2 - Some Correct, Some Wrong; 1 - Nothing Happened, 0 - 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 the ten participants. Figure 2 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 2 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, or 3 training points (see Figure 3). 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,

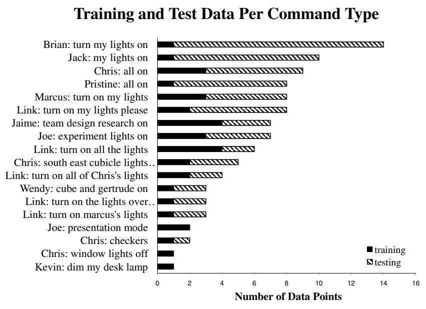

Figure 2. The number of training and test data points collected for each group of similar commands labeled with representative instances.

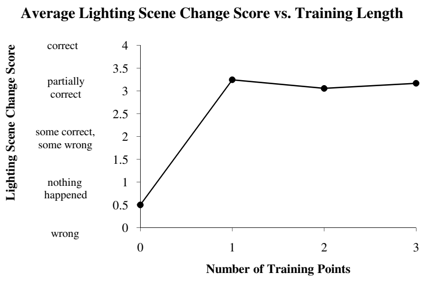

Figure 3. The test point score as reported by the participants versus the training length for the specific type of command.

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?" eight out of ten participants responded *Yes*, and two responded *Maybe*—the options were *Yes*, *Maybe* and *No*.

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 at least similar to the performance during the study.

RELATED WORK

Controlling lighting and other devices in the home is not new, nor is using speech to do so, but using speech to control configurations of devices is novel. Quesada *et al.* address the interface challenge in the home machine environment [11] 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.* [9] 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.

In recent years, alternative factorization methods such as least-squares NMF (Non-negative Matrix Factorization) have found favor over singular value decomposition (SVD) in cases where patterns are not orthogonal. This choice is especially true when the patterns appear "non-negatively," which is indeed the case for both light intensities and for word frequencies in user commands. Thus, NMF seems like a natural candidate for SNAC analysis. Furthermore, NMF has been shown to be superior to SVD for pure text clustering [6] or for image segmentation [6], which is similar to our light grouping problem.

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 to which their commands refer.

REFERENCES

1. Adura Technologies. http://www.aduratech.com.
2. A. R. Chang and J. F. Canny. Illuminac: Simultaneous naming and configuration for workspace lighting control. Technical Report UCB/EECS-2008-119, 2008.
3. N. Dahlbäck and A. Jönsson. Wizard of Oz studies – why and how. In *IUI*, 1993.
4. S. Escuyer and M. Fontoynont. Lighting controls: a field study of office workers' reactions. *Lighting Research and Technology*, 33(2), 2001.
5. 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, 2003.
6. D. D. Lee and S. H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755), 1999.
7. J. Love. Field performance of daylighting systems with photoelectric controls. In *Right Light Three, Energy Efficient Lighting*, 1995.
8. T. Moore, D. J. Carter, and A. I. Slater. A field study of occupant controlled lighting in offices. *Lighting Research and Technology*, 34(3), 2002.
9. M. C. Mozer. Lessons from an adaptive house. In *Smart environments: Technologies, protocols, and applications*. 2005.
10. 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*, 1997.
11. 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, 2001.
12. M. Weiser and J. S. Brown. The coming age of calm technolgy. In *Beyond calculation: the next fifty years*. 1997.