User-Extensible Natural Language Spoken Interfaces for Environment and Device Control

Ana Ramirez Chang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-162
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-162.html

December 17, 2008

User-Extensible Natural Language Spoken Interfaces for Environment and Device Control

by

Ana Ramírez Chang

B.S. (Carnegie Mellon University) 2002
M.S. (University of California, Berkeley) 2005

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:
Professor John Canny, Chair
Professor Maneesh Agrawala
Professor Kimiko Ryokai

Fall 2008

User-Extensible Natural Language Spoken Interfaces

for Environment and Device Control

Abstract

User-Extensible Natural Language Spoken Interfaces

for Environment and Device Control

by

Ana Ramírez Chang

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor John Canny, Chair

While the increase of electronic devices brings greater flexibility and control to our everyday environments, task-specific configuration of the myriad of devices becomes increasingly burdensome to the user. Ideally, interfaces for collections of such devices in our everyday environment should be calm, easy to learn and use, support control of favorite task specific configurations and allow the use of familiar names for configurations without learning. We propose a user-extensible natural language spoken interface for environment and device control. In order to support user customization in a natural language interface, the system must simultaneously learn the configurations and names for those configurations. We call this the simultaneous naming and configuration (SNAC) problem.

In this thesis, we identify and describe an approach to the simultaneous naming and configuration problem, and present a specific realization of our approach to the SNAC problem called Illuminac. We find that this kind of user-extensible speech interface is quite effective for some types of users and thus an important component in a complete solution to device configuration. In particular, we observe that users are willing to train the system, able to remember their commands, and willing to use voice commands in many cases.

Professor John Canny
Dissertation Committee Chair

To my husband Evan and parents Jeannette and Flavio

for their continual love, support and encouragement.

Contents

Contents **ii**

List of Figures **iv**

List of Tables **vi**

Acknowledgments **vii**

1 Introduction **1**
- 1.1 Background . . . 2
- 1.2 Thesis . . . 5
- 1.3 Contributions . . . 5
- 1.4 Outline . . . 6

2 Workspace Lighting Control **7**
- 2.1 User-Extensible Speech Interface for Workspace Lighting Control . . . 9

3 Related Work **11**
- 3.1 Natural Speech Interface Design . . . 11
- 3.2 Smart Homes and Home IT . . . 12
- 3.3 Workspace Lighting Control . . . 13
- 3.4 Multimodal Interfaces . . . 14
- 3.5 Applications of Matrix Factorization . . . 15

4 Approach to SNAC Problem **16**
- 4.1 Traditional Approach to Designing Speech Interfaces . . . 16
- 4.2 Simultaneously Discovering Naming and Configuration . . . 17
- 4.3 Learning Model . . . 18

5 Illuminac **20**
- 5.1 Illuminac from the User's Perspective . . . 20
  - 5.1.1 Training Mode . . . 21

5.1.2 Running Mode 24
5.1.3 Overlapping Configurations 25
5.2 Workspace Details 25
5.3 Illuminac Implementation 27
5.3.1 Learning Model 27
5.3.2 Data Representation 30
5.3.3 Automatic Speech Recognition 35
5.3.4 System Architecture 36

6 Illuminac Iterative Design and Evaluation 41
6.1 Methodology 41
6.1.1 Iterative Design 41
6.1.2 Participant Pool 42
6.1.3 Study Style 42
6.2 Overview of Iterative Design Steps and Evaluation 43
6.3 Lo-Fidelity Text-Based Data Collection 44
6.4 High-Fidelity Formative Data Collection 46
6.4.1 Data Collected 46
6.4.2 Results 47
6.4.3 Discussion 58
6.5 First Live Deployment 59
6.5.1 Study Description 59
6.5.2 Data Collected 63
6.5.3 Study Results and Discussion 63
6.6 Second Live Deployment with Graphical User Interface Control 65
6.6.1 Methodology 65
6.6.2 Study Details 67
6.6.3 Results 70
6.6.4 Discussion 79

7 Conclusion 81

Bibliography 84

A First Live Deployment Post Questionnaire 89

B Second Live Deployment Questionnaires 91
B.1 GUI Phase Post Questionnaire 91
B.2 Illuminac Phase Post Questionnaire 92

C Stop Phrase List 94

List of Figures

5.1 Next to each microphone there is a small display that contains an LCD, four vertical LEDs and five buttons. The LEDs show the volume of the microphone (a). The LCD shows "processing . . . " if the system is processing an audio command (a) and then shows the output of the automatic speech recognizer (b). The user can undo the effect of the last command if necessary by pushing the button labeled "x" (c). 22
5.2 The Illuminac training interface with the three steps to train the system on a configuration and command marked. 23
5.3 Tania and Ashley's lighting configurations at their desks overlap. If Ashley turns off her lights while Tania still has hers on, only the non-overlapping lights in Ashley's configuration will be turned off. 25
5.4 Picture and floor plan of the workspace for which we have implemented a user-extensible natural speech interface for the lighting control. Each of the 79 individually-controllable lights as well as the eight microphones are shown. 28
5.5 Web-based graphical user interface used to manually configure the array of lights in the workspace. The button used to enter training mode is highlighted. 29
5.6 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 that are given are in black, and matrices that we calculate are in gray. 34
5.7 Illuminac system architecture and information flow for each of the tasks: Using Illuminac, Training Illuminac and using the GUI. 37

6.1 Web-based graphical user interface used to manually configure the array of lights in the workspace when the wizard was not available. 45
6.2 Number of commands recorded per participant. 47
6.3 Number of commands recorded per group of similar commands, with a representative command shown for each group. 48
6.4 Sample groups of commands 49

6.5 Cross validation accuracy rate as a function of training length for data with manual and automatic speech recognition transcripts. Lef: accuracy as a function of average training length. Right: accuracy as a function of max training length. 51
6.6 ROC graph for a model trained on a maximum of five commands per group, and an average of 1.8 commands per group. 53
6.7 The first version of the training graphical interface. 61
6.8 The number of training and test data points collected for each group of similar commands. The command group labels are representative commands recorded by the participants. 62
6.9 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 64
6.10 *Top:* Quantitative results from the mixed phase. The percentage of times each participant used Illuminac versus the GUI. *Bottom:* A high level description of the participants experience with Illuminac based on the questionnaires and exit interviews. 72

List of Tables

6.1 The study contained three phases: GUI–participants used the GUI exclusively; Illuminac–participants used Illuminac exclusively; Mixed–participants used GUI and Illuminac as they saw fit. 69
6.2 Representative command for each lighting configurations users trained the system on and the number of training points provided for each configuration. Note: users were asked to state their name at the beginning of the command to simulate speaker identification. 75

Acknowledgments

Thanks to my advisor, Professor John Canny, for his invaluable guidance on the research.

Thanks to my husband, Evan Chang, for his unconditional love, support and encouragement. And for doing the grocery shopping and cooking while I was working on my paper deadlines.

Thanks to my mom, Jeannette Peterson, for teaching me by example to be a an independent thinker. To my dad, Flavio Ramírez, for introducing me to computer science. To my parents for their love, support and encouragement. To my entire family, thanks for respecting my deadlines, not all families understand that.

Thanks to my very good friends Josephine Chang and Divya Ramachandran for being there in the ups and downs and all the times in between.

Thanks to my lab mates in the Berkeley Institute of Design for their feedback on my research, for being guinea pigs with the Illuminac system, and for participating in all the user studies.

Thanks to my advisors for my masters project, Marc Davis and Jennifer Mankoff, for getting me off to a good start in research.

Thanks to the members of WICSE for helping create a supportive graduate school environment. Thanks to Sheila Humphreys for helping maintain connections between WICSE members, new and old, and for starting and supporting WICSE. Thanks to La Shana Polaris for helping me navigate the bureaucracy at Berkeley.

Chapter 1

Introduction

In this dissertation, I study the challenge of controlling large sets of devices, such as, lights, blinds, televisions, speakers, or projectors, in ubiquitous computing environments like the home or office. I propose an approach to these challenges and study a realization of that approach. The realization is a system we call Illuminac which is an:

- open vocabulary / user extensible,
- natural speech interface that requires a
- moderate amount of learning.

The system is designed to allow users to configure a large array of individually controllable lights in a big, shared workspace. I present the iterative design steps we used in designing and implementing Illuminac and a summative evaluation comparing the system with a state-of-the-art web-based graphical user interface.

1.1 Background

The number of electronic devices in our environment is ever increasing. While this trend brings greater flexibility and control, configuring each individual device becomes ever more tedious. For example, in the workplace, to set up a meeting, one must dim certain lights and configure the appropriate input for the projector. In the home, to prepare the environmental state for cooking, one might want to switch the radio on and tune it to the news, turn the volume of the speakers up, and make the kitchen lights bright. Then, to prepare the environmental state for watching television, one might dim the lights in the living room, switch 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 projector, 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. That is, the user should only have to specify an intuitive *name* of the environmental state rather than the *configuration* of each individual device needed to achieve the desired state. 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 (e.g., a key press or a spoken command). In the home example, the user should be able to say "prepare the room for cooking dinner" or "I'd like to watch TV now" rather than specifying the configuration of the radio, speakers and lights to achieve the cooking or television watching environmental state. The environmental state could be controlled automatically in a "smart-home" environment if the user's current activity could be inferred, but even with this kind of system, a conscious, foreground type of interface is necessary to control the environmental state when the intelligent system does not prepare the environment appropriately or is not able to infer the user's current activity.

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 and Brown [1997]). 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.

A truly calm interface should not require users to learn a new command language. In the home example, the user should be able to say "I'd like the cooking mode please," "apply the mode for preparing dinner," or a similar variation and have the system give a similar response. They should also be able to say "reading mode" or "TV mode" 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. We propose a user-extensible spoken language interface to allow the interface to be customized to the users' usage patterns. The novel aspect is that the system learns configurations and names simultaneously to support the specific configurations used by the users and the names for those configurations most natural and easy to remember for those users.

This problem cannot be solved by simply memorizing command strings and the appropriate device settings. In this 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. For instance, there will be many training strings for watching television that include the word “TV” and many other filler words, but which all specify a similar light and window shade pattern. Since the system looks for common patterns in names and configurations, the word “TV” will be strongly present in a pattern that includes television watching settings. Thus it is able to infer that “TV” is a salient keyword for watching television versus “please” or “now” that may also occur in command strings. Similarly, “movie” and “TV” will typically occur with similar patterns of devices, and the system will be able to infer that they are aliases in this context. We call 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 a similar challenge. In the home, a senior or disabled resident could use a standard cellular phone or cordless phone, which is easily kept within easy reach, to prepare the environment in another room before moving into it and to turn off the necessary devices in a room after leaving it (e.g., the lights, television and speakers). In this dissertation, we study a shared workspace environment with a large number of individually controllable lights. This case is expanded upon in the next chapter.

1.2 Thesis

My thesis is that a user-extensible, natural language spoken interface is an easy to learn and use approach to controlling configurations of devices in an environment such as, the workplace or the home. It is an important piece of a complete solution to controlling configurations of devices which may also include a graphical interface, a physical interface and/or an automated approach.

1.3 Contributions

In this dissertation, I 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 that arises in natural environment control. Our approach results in a calm interface that learns intuitive *names* for commonly used *configurations* of devices in a home or work environment instead of requiring the user to learn hard coded, static names for fixed configurations of devices, or worse, hard coded names for individual devices (Chapter 4).

- We identify an appropriate learning algorithm for the simultaneous naming and configuration problem: non-negative matrix factorization (NMF) (Section 4.3).
- We show the applicability of our approach to the SNAC problem on natural speech interfaces for workspace lighting control by implementing (Chapter 5) and deploying (Section 6.5) 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.
- We show users are willing and able to train the system to work for their favorite configurations, able to remember their commands, and prefer to use voice commands over using a graphical user interface to control the lighting scene in many cases. We believe these results should transfer to other task-specific control of a collection of devices in other environments such as the home (Section 6.6).

1.4 Outline

In Chapter 2, I describe the domain in which we designed and deployed Illuminac, workspace lighting control. In Chapter 3, I ground the work within related work. Chapter 4 presents our approach to the SNAC problem and explains why we chose to use non-negative matrix factorization as the learning model. I describe the Illuminac system in Chapter 5 including how a user trains and uses the system and how the system is implemented. Chapter 6 outlines the iterative design process we followed in designing and implementing Illuminac, as well as the deployment and studies of Illuminac.

Chapter 2

Workspace Lighting Control

Before I present my approach to the SNAC problem and the Illuminac system, I describe the sub-domain for which Illuminac was 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. 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 [Moore et al. 2002].

Proprietary granular lighting control systems for both retrofit and new construction have been available for some years [Adura Technologies]. Most recently, a low-cost industry standard has emerged: DALI (Digital Addressable Lighting Interface [DALI]) 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 did not mind the occasional error in an automatic lighting control system as long as they had an easy, manual way to correct the lighting scene [Escuyer and Fontoynont 2001]. A study by Love showed that automatic controls were commonly disabled by users [Love 1995]. 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 user-extensible 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 "can I get my cube lights on?" to turn on the four lights around her desk.

2.1 User-Extensible Speech Interface for Workspace Lighting Control

As alluded to earlier, to support customized speech commands for personalized lighting configuration, we must address the SNAC problem. Specifically, we need to discover the lighting scenes commonly used in the space and the names for those lighting scenes. A lighting scene is a set of lights and an intensity setting for each of the lights in the set. A name for a lighting scene is a set of words. For example, a user might have two intensity settings for the three lights over his desk—one for desk work and one for computer work. He might say "Please turn Jason's reading lights on" or "can I get Jason's computer lights on?" The desk work lighting scene would be the three lights over Jason's desk set to 100% intensity and the computer lighting scene would be the three lights over his desk set to 50% intensity. The name of the desk work lighting scene might be {Jason's, reading} and the name of the computer lighting scene might be {Jason's, computer}.

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, implementation and evaluation of Illuminac are discussed in Chapter 5 and Chapter 6, in the remainder of this section, we sketch how the user interacts with our system.

In order to customize the Illuminac system to work with their specific light configurations and configuration names, the users train the system. They need only provide one to two training points for each configuration they use to get good results. In our experience, users have between one and five different configurations they use regularly. The users can add training points all at once when they begin using the system or one at a time as needed — such as when they come across a configuration they have not trained the system on or if they try a command and do not quite get the correct configuration.

To add a training point for a particular light configuration the user needs to record her name, command, and configuration. 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 points, 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.

Chapter 3

Related Work

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

3.1 Natural Speech Interface Design

Much of the work in speech interfaces has been for systems where all of the objects are known *a priori*. When designing these kinds of systems, the designer can use the wizard-of-Oz technique [Dahlbäck and Jönsson 1993] to support names for those objects that are natural to the users. For example, in the RoomLine [Bohus] system, a speech interface that allows users to reserve rooms over the phone, all of the rooms, the sizes of the room, and the equipment available in the rooms were known at the time the speech interface was designed. Such speech interfaces can be grammar-based like many successful commercial systems (e.g., BeVocal [Nuance], Tellme [TellMe Networks]), they can be statistical language model-based [Brown et al. 1990], or they can use a combination of both [Rayner et al. 2004]. In the case of designing a speech interface for controlling configurations of devices, the objects (configurations), are not known *a priori.* Since we would like the system to support configurations that are used in the space instead of preset, possibly artificial configurations, in addition to supporting natural names for those configurations, we propose a user-extensible system. The user-extensible system must therefore be trained by the users on both their configurations as well as their names for those configurations.

Early work in speech interfaces for physical spaces (the "Put-That-There" work) did allow users to train a speech system to work on their customized names for objects [Bolt 1980]. The system allowed users to manipulate simple drawing shapes such as circles and squares in a large display by using a combination of speech and gesture. Users were able to use generic names for the shapes, or train the system to recognize customized names for specific object instances. The system, however, did not support customized names for sets of objects, it only supported a one-to-one mapping from customized names to individual objects.

3.2 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 [Quesada et al. 2001] 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. [Mozer 2005] 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 [Juster and Roy 2004]. 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, as it is designed to support a static set of lighting configurations. We focus on the speech interface and supporting customized commands and configurations for each new set of devices without having to have a designer perform the customization.

3.3 Workspace Lighting Control

Commercial as well as research intelligent workspace lighting control systems exist, but they do not address the interface for manual control when users wish to override the system. Much of the work focuses on figuring out to what intensity to set the lights based on a variety of constraints, such as the amount of daylight present, the user's preferences and the building manager's requirements [Wen et al. 2006; Singhvi et al. 2005]. Whether a manual interface is used in conjunction with an intelligent system (when the users want to override the system) [Escuyer and Fontoynont 2001] or a manual interface is used instead of an intelligent system (some users disable intelligent systems [Love 1995]), the manual interface should be natural and easy to use. If it is not easy to use, users will be less likely to use it, either resulting in a waste of energy (the default intensity is too high), or a potential decrease in productivity (the default intensity is too low). In Illuminac, our focus is on designing a natural, easy-to-use manual interface for the lighting control. Illuminac can be used on its own, or in conjunction with an intelligent lighting control system.

3.4 Multimodal Interfaces

It is well established that speech and gesture work well together [Mcneill 1992; Bolt 1980]. Wilson's work on the XWand [Wilson and Shafer 2003] demonstrates a system where 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 that "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 that pointing is possible without the cameras, but it 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 cannot be predetermined and can also be used in cases where the calibration and installation of cameras is not possible.

3.5 Applications of Matrix Factorization

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 [Lee and Seung 1999] or for image segmentation [Lee and Seung 1999], which is similar to our light grouping problem. Barnard et al. match words and pictures [Barnard et al. 2003] using an “aspect” probabilistic model, which is another type of factor model. Other candidate factor models include Latent Dirichlet Analysis (LDA) [Blei et al. 2002] and GaP [Canny 2004]. 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 [Lee and Seung 1999].

Chapter 4

Approach to 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 (1) the commands natural to the users, and (2) the configurations naturally used in the domain.

4.1 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 [Dahlbäck and Jönsson 1993] to gather formative data including sample utterances, which help 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 by 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.

4.2 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 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.

In order 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 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 perform the customization.

4.3 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), a standard learning algorithm, 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 each light to only 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; specifically they

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

Thus, we are simultaneously 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.

Chapter 5

Illuminac

In this chapter I present the system we designed and deployed to study our approach to the SNAC problem. We describe the system from the user's perspective—how to use the Illuminac system, the space in which Illuminac was deployed, the implementation details of the system and the Illuminac system architecture. In the next chapter we describe our iterative design process and summative evaluation studies.

5.1 Illuminac from the User's Perspective

Illuminac is a user-extensible natural language speech interface for controlling configurations of lights (individual settings for a collection of lights) with natural speech commands. For example, Ashley might say "⟨Ashley⟩ Can I get my desk lights on please" to turn on the three lights over her desk to 75% of full intensity.

Users train the system on their personalized lighting configurations and commands and can then speak their commands into any of the eight microphones around the lab.

Each microphone has a feedback display next to it (shown in Figure 5.1) that shows the microphone volume and the transcript from the speech recognizer.

5.1.1 Training Mode

To add a training example, the user records her command and configuration through a web-based graphical interface (shown in Figure 5.2). To do so, she completes the following three steps:

1. record her command using one of the microphones around the room (prefaced by her name to simulate speaker identification);
2. correct any recognition errors in the automatic speech recognition transcript using the web-based graphical user interface; and
3. record her desired lighting scene using the web-based graphical user interface to manually set the lighting configuration.

The user uses a microphone to record her command instead of simply typing her command into the web page to increase the probability she is recording a command she would say, and not one she would type. The user corrects any recognition errors so that an incorrect command is not added to the speech recognition language model. The corrected command is not used to retrain the acoustic model, that is kept constant (Section 5.3.3 describes the corpus we use to train the acoustic model.) The web interface allows the user to "paint" on an intensity from 1 to 9, it does not allow the user to paint on an intensity of 0, and explains at the top of the page that the system should not be trained on "off" commands. Instead the off commands are derived from the "on" commands.

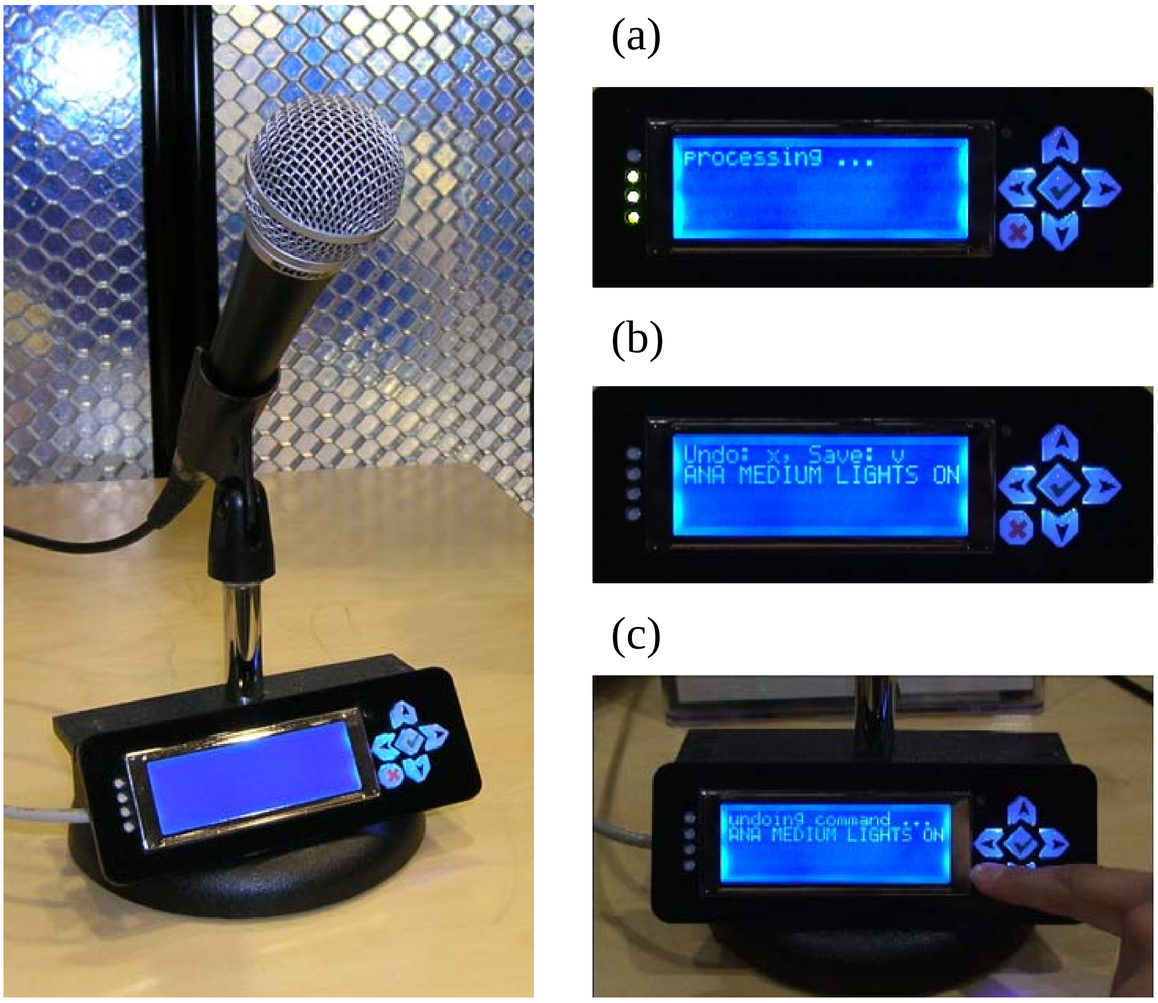


Figure 5.1: Next to each microphone there is a small display that contains an LCD, four vertical LEDs and five buttons. The LEDs show the volume of the microphone (a). The LCD shows “processing . . . ” if the system is processing an audio command (a) and then shows the output of the automatic speech recognizer (b). The user can undo the effect of the last command if necessary by pushing the button labeled “x” (c).

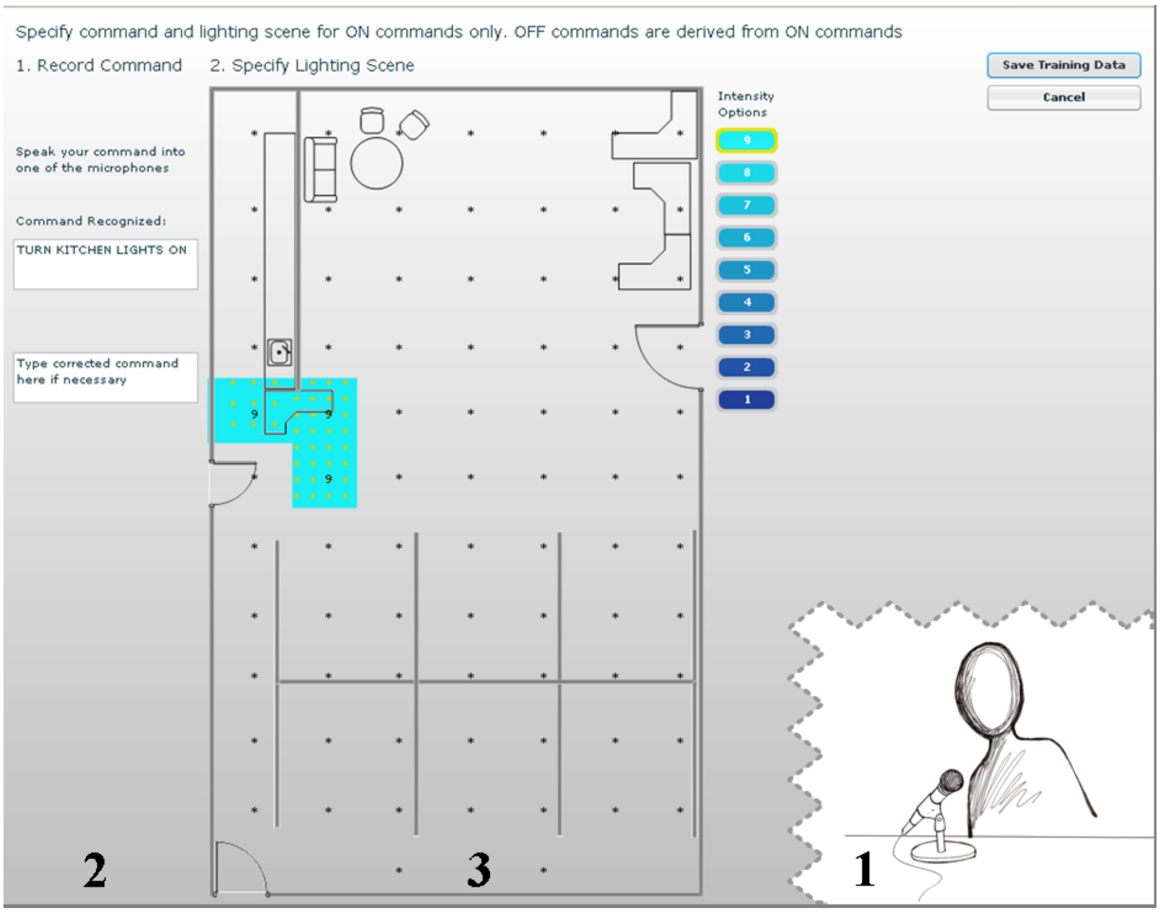

Specify command and lighting scene for ON commands only. OFF commands are derived from ON commands
1. Record Command
2. Specify Lighting Scene
Save Training Data
Cancel
Speak your command into one of the microphones
Command Recognized:
TURN KITCHEN LIGHTS ON
Type corrected command here if necessary
Intensity Options
9
8
7
6
5
4
3
2
1
2
3
1

Figure 5.2: The Illuminac training interface. To train the system on a configuration and command, the user completes the following three steps: (1) record her command using one of the microphones around the room (prefaced by her name to simulate speaker identification); (2) correct any recognition errors in the automatic speech recognition transcript using the web-based graphical user interface; and (3) record her desired lighting scene using the web-based graphical user interface to manually set the lighting configuration. (The drawing of the user recording her command is by Lora Oehlberg.)

When interpreting a command, the system first decides if it is an "on" command or an "off" command. If it is an "on" command, the system applies the new intensity setting as estimated from the learning model. If it is an "off" command, the system sets the intensity of the affected lights to 0. This is to ensure the intensity of the lights are set to 0, and not a low intensity close to 0 when the lights are being turned off.

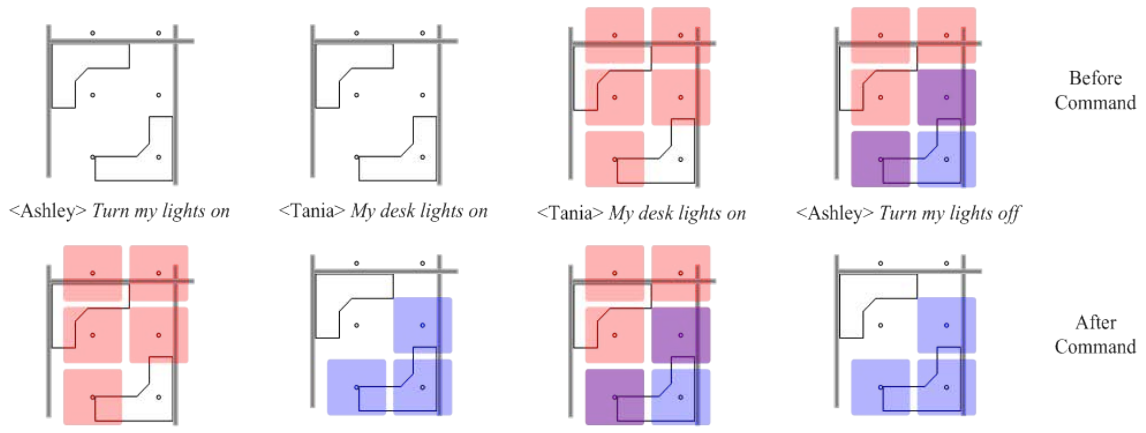
After the user adds a new training point, the system retrains the model on the old training data plus the new data point. It also adds the command to the speech recognition language model to increase the recognition accuracy for the command the next time it is used. The new model is trained and ready 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.

5.1.2 Running Mode

In the running mode, a user can use any 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 estimated lighting scene.

While the system is transcribing a command, the text "processing ..." is displayed in the GUI and a small LCD display next to each microphone (typically takes about two seconds). This lets the user know the system "heard" them and is processing the command.

The automatic speech recognition transcript is shown on the small LCD display next to each microphone. If the estimated lighting scene is incorrect, the user can undo the



<Ashley> *Turn my lights on*
<Tania> *My desk lights on*
<Tania> *My desk lights on*
<Ashley> *Turn my lights off*
Before Command
After Command

Figure 5.3: Tania and Ashley's lighting configurations at their desks overlap. If Ashley turns off her lights while Tania still has hers on, only the non-overlapping lights in Ashley's configuration will be turned off.

last change by pressing the "x" button next to the small LCD screen.

5.1.3 Overlapping Configurations

Some of the lighting configurations in the workspace may overlap. For example, users who share a cubicle often have overlapping lighting configurations over their desks. If two overlapping configurations are on and one of the cubicle-mates turns off her lights, the system will only turn off the lights that do not overlap with the other cubicle-mate's lighting configuration. Figure 5.3 shows and example where Tania and Ashley's lighting configurations overlap.

5.2 Workspace Details

Illuminac was designed for and is deployed in a 2,300 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 (i.e., 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. The multi-use space has a presentation screen, a "soft space" with a couch and chairs, and a set of four computers for visitors of the lab to use. There is also a tool shop in one corner of the room. Figure 5.4 shows a picture and the floor plan of the room.

The room has 79 compact fluorescent lights mounted overhead. Each light has two controls: one to control the intensity and one to turn power on and off. The intensity control uses custom microcontrollers operating 0-10V dimmable fluorescent ballasts. The intensity of each light can be controlled independent of the other lights, and all light intensities can be updated several times per second. The power controls are less granular, the lights are divided into banks of two to seven lights that are turned on and off together. This is an artifact of the hardware available when we retrofitted the space. We used X10 to control the power of each light. The X10 signal is too slow to control such a large number of devices individually, which is why we control the power in small banks of lights. With the new DALI [DALI] standard, both the power and the intensity of each light can be controlled fast enough to be controlled individually.

The intensity of each light and the power for each bank of lights can be controlled over the network via a web interface (shown in Figure 5.5). Occupants of the lab can control the lighting via the web interface from their personal computers or from one of the public machines. The public machine next to the entrance (labeled "Main Public Machine" in Figure 5.4) always has the web interface open.

The room is equipped with eight desk microphones throughout the room to allow easier access to the Illuminac system, through which users can control the lighting. 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.

5.3 Illuminac Implementation

5.3.1 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 [Lee and Seung 1999]. 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

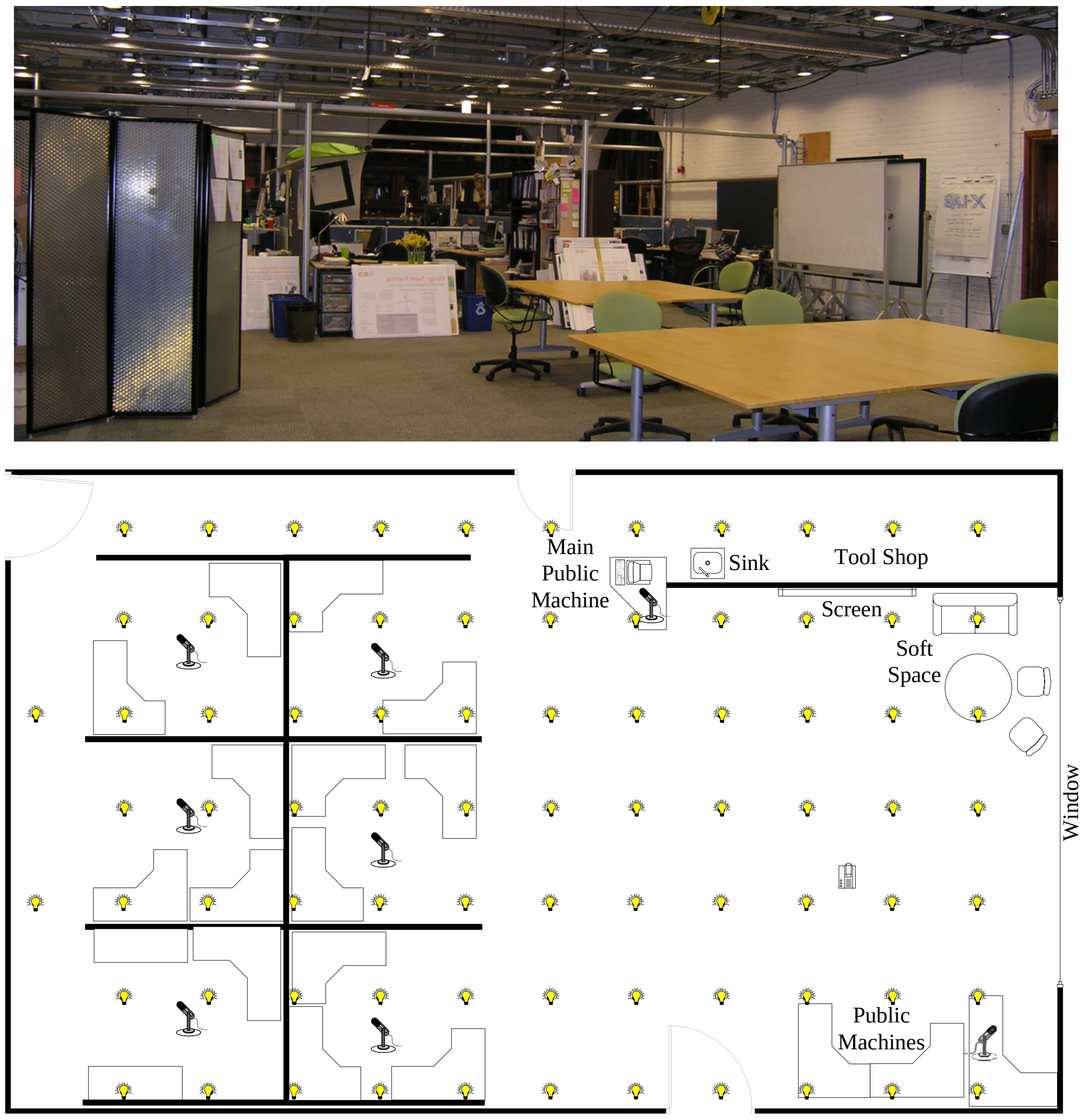
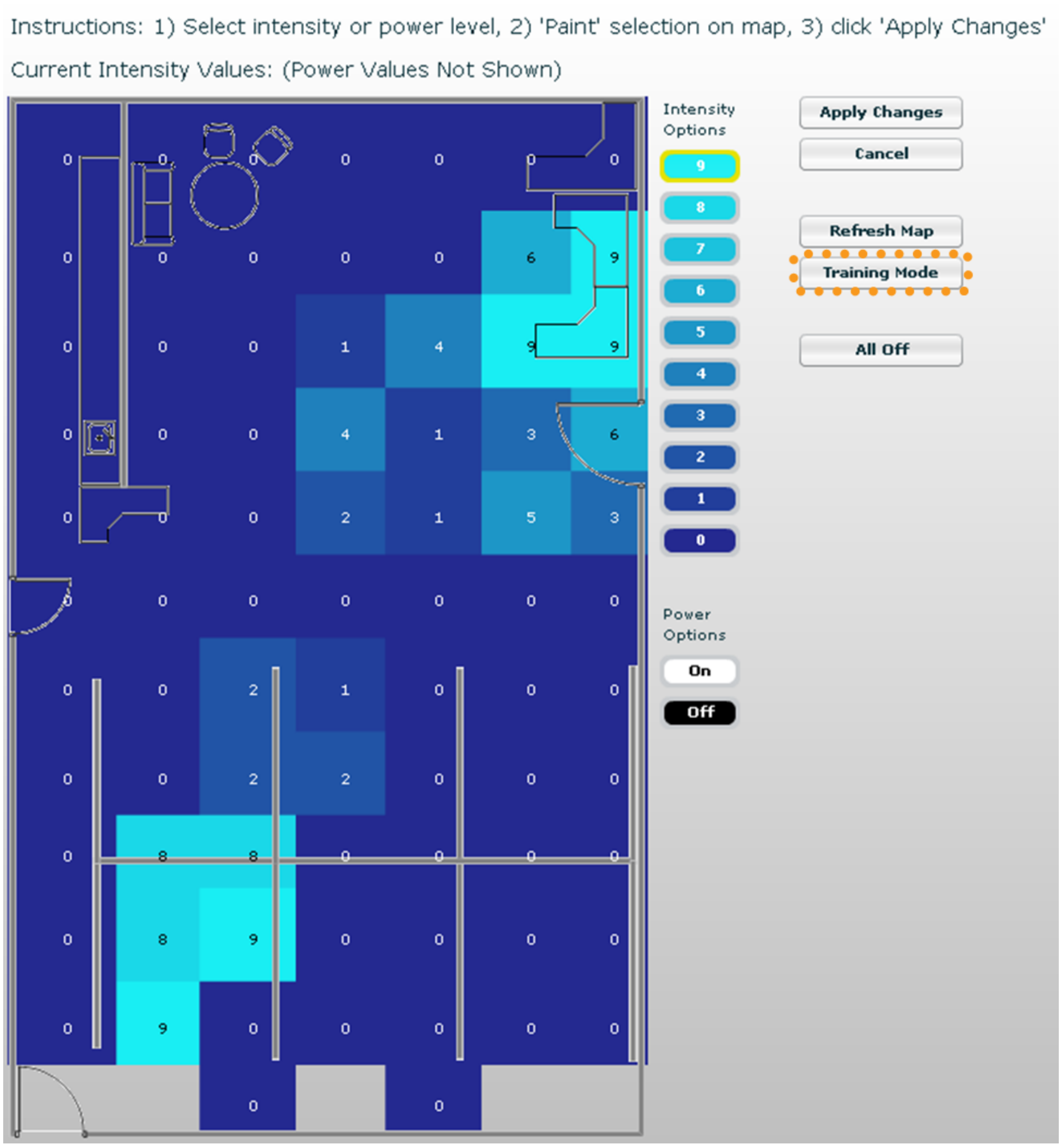


Figure 5.4: Picture and floor plan of the workspace for which we have implemented a user-extensible natural speech interface for the lighting control. Each of the 79 individually-controllable lights as well as the eight microphones are shown.



Instructions: 1) Select intensity or power level, 2) 'Paint' selection on map, 3) click 'Apply Changes'

Current Intensity Values: (Power Values Not Shown)

Figure 5.5: Web-based graphical user interface used to manually configure the array of lights in the workspace. The button used to enter training mode is highlighted.

NMF to learn which lights should change given a new command and the current lighting scene.

5.3.2 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. Before calculating the term-frequency vector, we remove any stop phrases—generic words or phrases not relevant to the configuration names. Thus the length of the vector is the number of unique unigrams or bigrams in all of the commands not including stop phrases. 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."

We derived the stop phrase list from the training data we collected in a formative training data collection study (described in Section 6.4). Appendix C lists the stop phrase list we used. In practice, the stop phrase list must be created for each domain, such as workspace lighting control or smart home control. The stop phrase list should transfer to other system within the same domain. For example, our stop phrase list should transfer to another workspace with different users. If commands in a different domain tend to be more complex than the phrases in our domain, a simple grammar could also be used to remove unrelated words or phrases from the commands.

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.

Users do not train the system on "off" commands, instead they are derived from the "on" commands. This is because we want to make sure the lights are set to intensity zero when they are turned off, not just a low intensity. Based on data from the formative data collection studies, the words "off" or "out" are added to the "on" command to derive the "off" command. Note, "on" commands do not need to have the word "on" in them, but to turn the lights off, the command must have the word "off" or "out" in it (which was always true in our sample data). For example, if a user trains the system on the command "⟨Ashley⟩ Can I get my desk lights on please," the following corresponding "off" commands are also added to the language model used by the speech recognizer: "⟨Ashley⟩ Can I get my desk lights off please." and "⟨Ashley⟩ Can I get my desk lights out please." Note, users will not have to say one of those exact phrases, and those phrases may even be grammatically incorrect, but it ensures we keep the number of commands with "on," "off" and "out" roughly even in the language model so that both "on" commands and "off" commands will be recognized by the speech recognizer and not just "on" commands.

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 [Lee and Seung 2000, 1999]:

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

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. 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 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 rate 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.

Normalization The matrices $\mathbf{U}$ and $\mathbf{V}^T$ in $\mathbf{X} \approx \mathbf{U}\mathbf{V}^T$ are not unique: multiplying the i^{th} column of $\mathbf{U}$ by s and the i^{th} row of $\mathbf{V}^T$ by $1/s$ produces the same product $\mathbf{U}\mathbf{V}^T$. To get a unique solution, we need to normalize one of the matrices and adjust the other one accordingly. In our case, we normalize matrix $\mathbf{U}$ by scaling the columns in $\mathbf{U}$ such that the maximum value in each column is 1. We then scale the corresponding rows in $\mathbf{V}^T$ so that the product $\mathbf{U}\mathbf{V}^T$ is preserved. In other words, we want all of the data points, the rows in $\mathbf{X}$, to have equal weight. We don't want some of the data points to be stronger than others.

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 5.6 depicts how to apply the trained model to a 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.

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

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

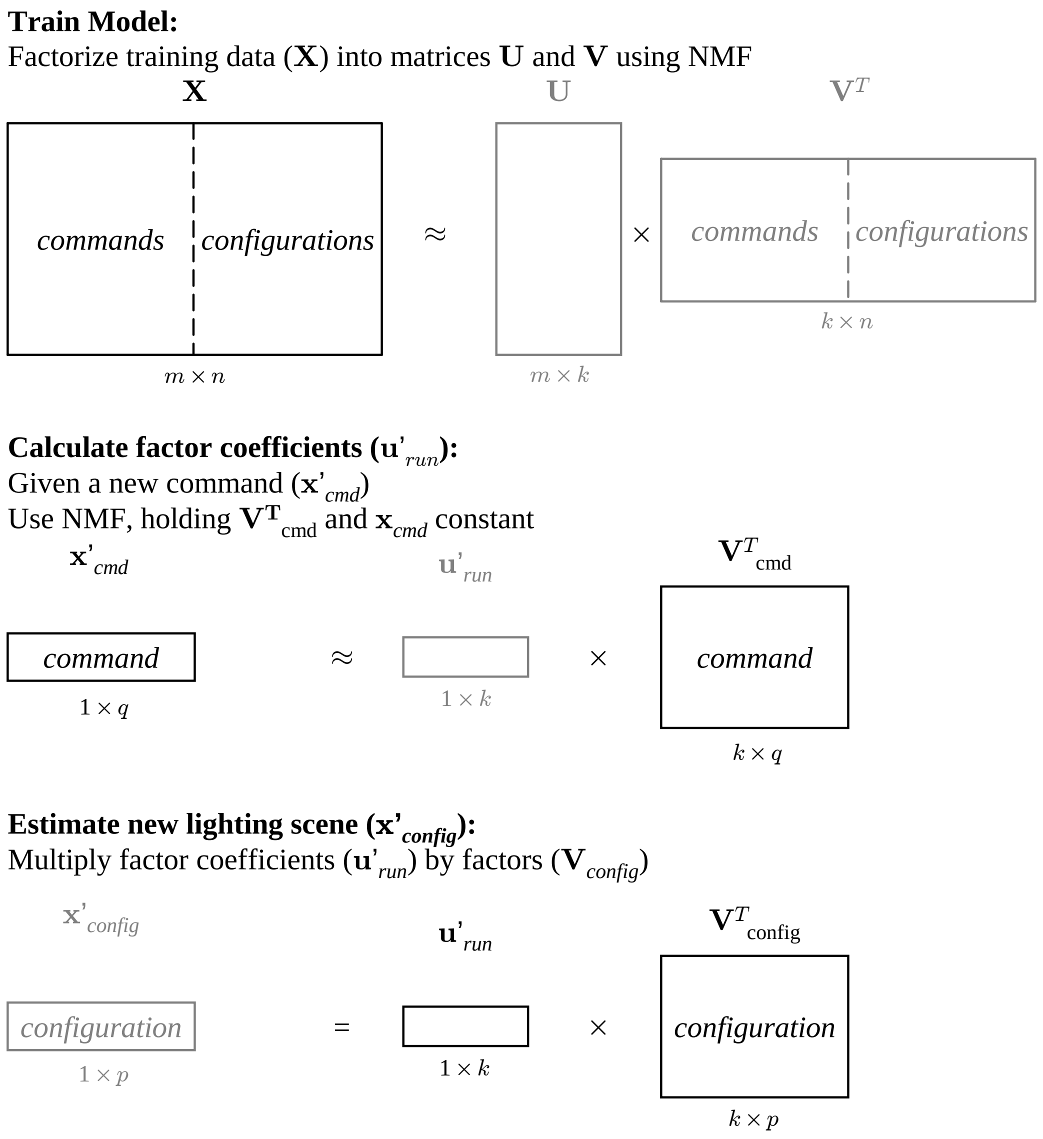

Train Model:
Factorize training data (X) into matrices U and V using NMF
X
U
V^T
commands
configurations
≈
×
commands
configurations
$k \times n$
$m \times n$
$m \times k$
Calculate factor coefficients (u'run):
Given a new command (x'cmd)
Use NMF, holding V^Tcmd and xcmd constant
x'cmd
u'run
V^Tcmd
command
≈
×
command
$1 \times q$
$1 \times k$
$k \times q$
Estimate new lighting scene (x'config):
Multiply factor coefficients (u'run) by factors (Vconfig)
x'config
u'run
V^Tconfig
configuration
=
×
configuration
$1 \times p$
$1 \times k$
$k \times p$

Figure 5.6: 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 that are given are in black, and matrices that we calculate are in gray.

$\mathbf{u'}_{run}$ 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 involved in the configuration, 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.

If the command is an "off" command, we set the intensity of the lights involved in the lighting scene to zero.

5.3.3 Automatic Speech Recognition

We are using Carnegie Mellon's Sphinx 3.5 speech recognizer [Placeway et al. 1997] to transcribe the commands. Sphinx 3.5 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 [Janin et al.], 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. 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 GB of RAM). As described above, the displays next to each microphone show the text "processing . . . " to let users know the system "heard" them while the recognizer is processing the command.

5.3.4 System Architecture

Illuminac is comprised of several sub-pieces written in various programming languages including C, C#, MATLAB® and Adobe® Flex® and ActionScript. In the following subsections I describe the different sub-pieces involved in processing a speech command, training the system on a new speech command and lighting configuration, and processing a lighting scene change made via the web-based graphical user interface. The complete system architecture and the flow of information for each of these tasks is shown in Figure 5.7.

Using Illuminac

When using the Illuminac system to change the lighting scene, the user speaks her command into one of the eight microphones around the room. All of those microphones are connected to a preamp and digitizer that connects the microphones to the computer. On

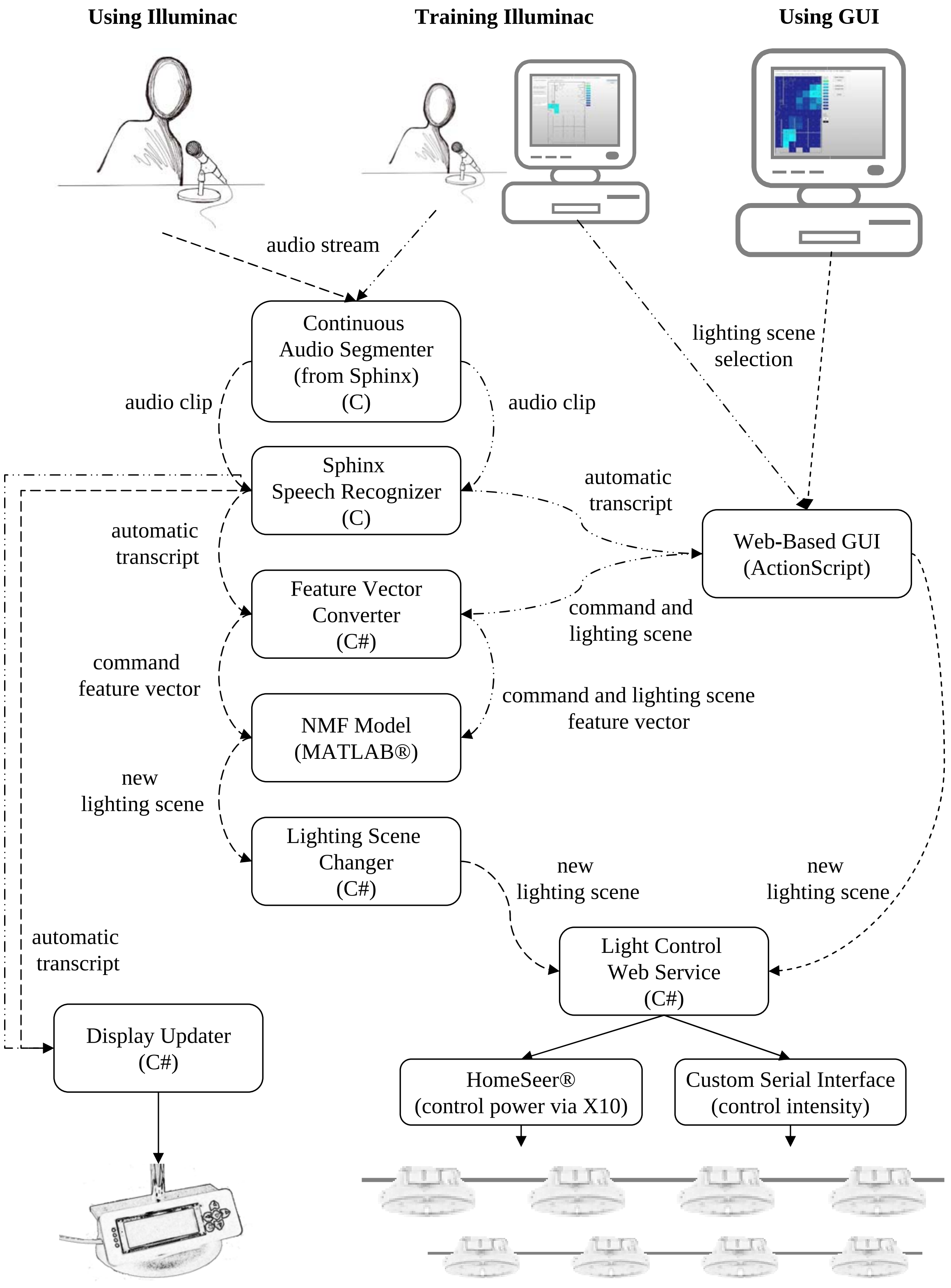

Using Illuminac
Training Illuminac
Using GUI
audio stream
Continuous Audio Segmenter (from Sphinx) (C)
audio clip
audio clip
lighting scene selection
Sphinx Speech Recognizer (C)
automatic transcript
Web-Based GUI (ActionScript)
automatic transcript
Feature Vector Converter (C#)
command and lighting scene
command feature vector
NMF Model (MATLAB®)
command and lighting scene feature vector
new lighting scene
Lighting Scene Changer (C#)
new lighting scene
new lighting scene
automatic transcript
Light Control Web Service (C#)
Display Updater (C#)
HomeSeer® (control power via X10)
Custom Serial Interface (control intensity)

Figure 5.7: Illuminac system architecture and information flow for each of the tasks: Using Illuminac, Training Illuminac and using the GUI.

the computer, a continuous audio segmenter (which comes with Sphinx) is used to segment the audio into utterances. The utterances are passed to the Sphinx speech recognizer that provides a transcript of the command. The transcript is then turned into a feature vector to which the NMF model is applied to get an estimate of the new lighting scene. The new lighting scene is passed to a web service that is connected to the lights hardware through a custom serial interface. The transcript of the command is also passed to a script that updates the displays next to each microphone with the transcript.

The script that passes the audio segments to the speech recognizer, computes the feature vector from the transcript, calls the code to apply the NMF model, and calls the web service with the new lighting scene is written in C#. The script that applies the NMF model to the feature vector and estimates the new lighting scene is implemented in MATLAB®. The lighting control web service is implemented in C# and runs on Microsoft's Internet Information Services web server (IIS). The script that updates the displays next to each microphone is also implemented in C#.

Training Illuminac

When training the Illuminac system on a personalized spoken command and lighting scene, the user enters the training mode by selecting "Training Mode" on the web-based graphical user interface. This calls a web service that keeps track of the current mode (training mode or running mode). Since the microphones are not tied to a specific computer or instance of the web-based GUI, the whole system is either in training mode or running mode. Since users don't change their lighting scenes very often throughout the day, there were very few conflicts of someone wanting to use the system in running mode while someone was adding a training point. This is also an artifact of the research system built and is not a limitation that would apply to another implementation of the system. This web service that keeps track of the current mode is not shown in the system architecture figure (Figure 5.7).

Once in training mode, when the user speaks her command into one of the microphones, the transcript is shown in the training page on the GUI and is passed to the script that updates the displays next to each microphone with the transcript. When the system is in training mode, the displays show "training mode ..." followed by the transcript. The user specifies the lighting scene using the GUI and makes any corrections necessary to the transcript shown in the GUI.

The lighting scene and corrected command are passed to the C# script that calculates the feature vector. The feature vector is added to the collection of training feature vectors, and the NMF model is retrained on this new collection of data. The new NMF model is stored in memory in the C# script that creates the feature vectors and calls a MATLAB® script that calculates the new NMF model. The training data is also written to a file that is used to load the NMF model into memory when the C# script is restarted.

The web-based graphical user interface used to add training points is implemented in ActionScript and was implemented using Adobe® Flex® Builder$^{\text{TM}}$. The web service that keeps track of the training/running mode of the system is implemented in C# and runs on the IIS web server.

Using the Graphical User Interface

The lighting control and training graphical user interfaces are part of the same web site. The lighting control GUI sends the new lighting scene specified by the user to the lighting control web service. The lighting control web service controls the intensity through a custom serial interface and the power settings on the lights through a commercial piece of software, HomeSeer®, that sends the appropriate X10 signals.

Chapter 6

Illuminac Iterative Design and Evaluation

In designing our user-extensible natural speech interface, Illuminac, we followed an iterative design process and performed two summative evaluation studies. In this chapter we outline our design methodology, describe each step in our iterative design process and present our evaluation studies.

6.1 Methodology

In this section we outline the methodology we followed in designing Illuminac.

6.1.1 Iterative Design

We used an iterative design process in designing Illuminac to ground our design and understanding of the SNAC problem in a real space with real devices, configurations of the devices used in practice and names for the configurations natural to the residents of the space. As in any iterative design, we started with a low-fidelity prototype and progressed with higher and higher prototypes until we were ready to deploy a live system. We continued to iterate once we had a live system and went on to deploy a second version of Illuminac.

6.1.2 Participant Pool

In each of our studies we recruited regular occupants of the space to participate. This was important to ensure the configurations and names for the configurations would be naturally occurring and not artificially created.

6.1.3 Study Style

We used an *in situ* study style in all of our studies. In this type of study the participants go about their regular business and schedule and only record their data when they want to change the lighting scene in the lab. We didn't tell our participants what configurations they should use or how they should be named. The participants were free to use any names and configurations they wanted to.

Similar to using residents of the space as participants, this type of study allowed us to collect data about configurations that were actually used and names for the configurations that the participants thought of at the time they wanted t change the lighting scene. This way the participants did not have to guess or remember what configurations and names they might use or used in the past.

6.2 Overview of Iterative Design Steps and Evaluation

Before we implemented support for audio, we began by trying a natural language text interface for the lighting control with a "wizard" and in some cases the users acting as the system and carrying out the lighting scene changes. This study allowed us to see if a natural language approach to the SNAC problem was a good idea and to get an idea of what kinds of configurations and names were naturally used in the space. (Section 6.3)

We followed the text-based study with a higher fidelity speech based study. In this study we had support to record audio commands but we did not have a live system yet. This study allowed us to collect high-fidelity training data that we used to design the learning algorithm. It also allowed us to understand the users' reaction to using speech to control the lighting scene. (Section 6.4)

Once we had confirmation form the users that they would like to use speech to control the lights and understood the properties of the SNAC problem with respect to the learning algorithm, we connected the audio recording, speech recognizer and learning algorithm to be able to deploy a live system. We evaluated the accuracy of the live system from the users' perspective and got a qualitative understanding of the usability of the system in through a short study with the deployed system. (Section 6.5)

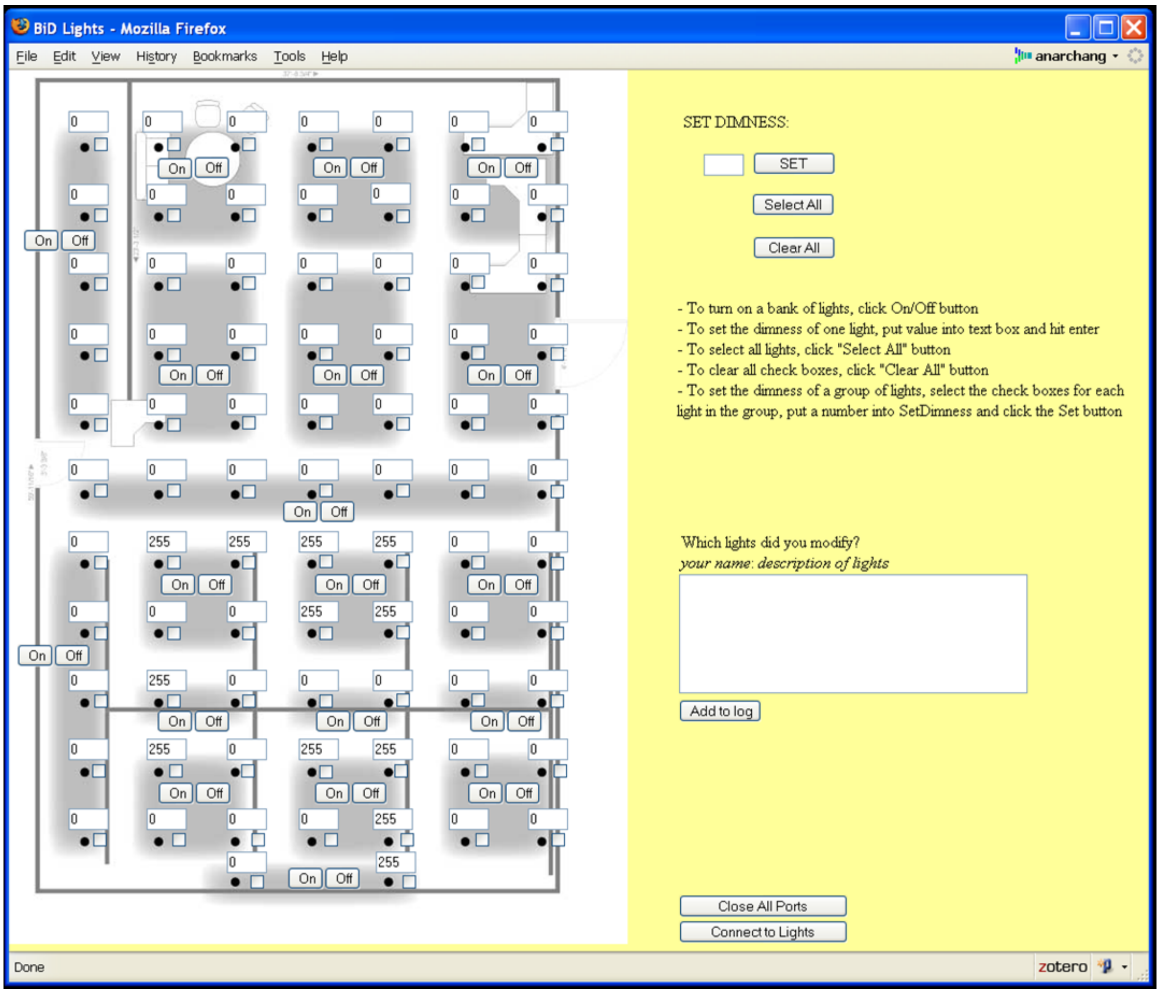
Finally, we conducted a longer study on the second version of Illuminac. In this study we focused on the user experience rather than the accuracy of the system. To do so, we compared the user experience with Illuminac and with a state-of-the-art graphical user interface. (Section 6.6)

6.3 Lo-Fidelity Text-Based Data Collection

We began our design of Illuminac with a low-fidelity wizard-of-Oz data collection step 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 based on the participant's message using the first version of the web interface. 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 (Figure 6.1). This way data could be collected even if the wizard was not at his desk.

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.



- To turn on a bank of lights, click On/Off button
- To set the dimness of one light, put value into text box and hit enter
- To select all lights, click "Select All" button
- To clear all check boxes, click "Clear All" button
- To set the dimness of a group of lights, select the check boxes for each light in the group, put a number into SetDimness and click the Set button

Which lights did you modify?
your name: description of lights

Figure 6.1: Web-based graphical user interface used to manually configure the array of lights in the workspace when the wizard was not available.

6.4 High-Fidelity Formative 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, which estimates 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; and
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.

6.4.1 Data Collected

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 (i.e., single

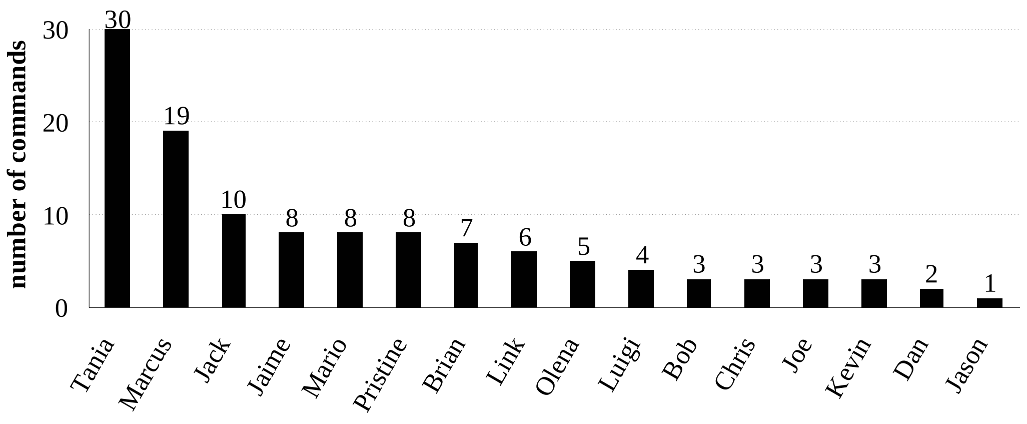


Figure 6.2: Number of commands recorded per participant.

words and two-word phrases).

Figure 6.2 shows the number of commands recorded by each participant. Commands were put into groups manually based on the lighting scene change and the words in the command, including the participant's name. Note, the commands are only grouped for testing purposes, they do not need to be grouped in the automated system. Figure 6.3 shows the number of commands in each group of similar command / configuration pairs. Figure 6.4 shows some examples of groups of commands.

6.4.2 Results

Quantitative Analysis

We evaluate our approach to SNAC for the workspace lighting control problem with a cross validation on the collected training data. The command / configuration tuples are manually divided into groups of similar tuples (shown in Figure 6.3 and Figure 6.4). We select one tuple from each group of similar tuples at random to hold out. The rest of

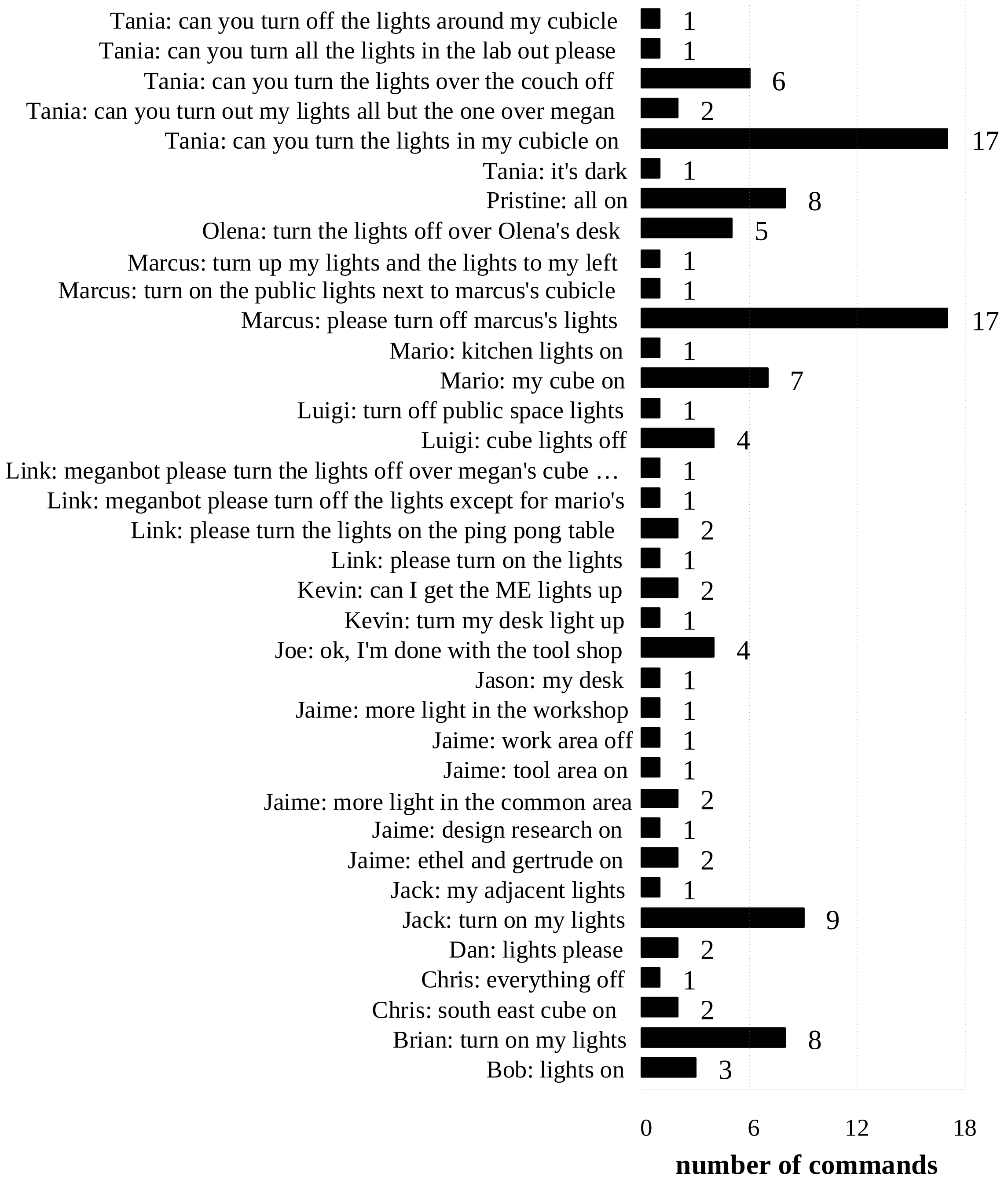


Figure 6.3: Number of commands recorded per group of similar commands, with a representative command shown for each group.

Mario:	my cube on
Mario:	cube off
Mario:	cube lights on
Mario:	cube lights on
Mario:	cube lights on

Tania:	can you turn the lights off the two lights over the couch
Tania:	can I get lights over the couch
Tania:	turn the lights over the couch on please
Tania:	can you turn the lights over the couch off
Tania:	can you turn the lights on above the couch
Tania:	can you turn the lights off over the couch

Link:	Joe needs some light in the tool shop
Joe:	ok I'm done with the tool shop
Joe:	can I get some lights in the tool shop please
Joe:	ok I'm done with the tool shop

Figure 6.4: Sample groups of commands

the tuples are used to train the model (we train one model total, not one for each group). To compare how well the model does as a function of training length, l tuples are selected at random from each group for $l = 1$ to 10 (excluding the tuple already selected as the test tuple). Since the number of tuples in each group varies widely, the number of tuples selected in practice is the minimum of l and the number of training tuples in a group. We plot the accuracy as a function of maximum training length as well as average training length in Figure 6.5.

The ROC graph convex hull area for the data tuples with manual transcripts ranges from 0.87 for an average and max training length of 1 to 0.88 for an average training length of 2.43 and a max training length of 10. These areas only dropped an average of 0.7 percent when an automatic speech recognizer with 30 percent word error rate was used to transcribe the audio clips. The NMF algorithm only took 0.89 seconds time to train on two weeks of data, which means the model can be re-trained after each new data point is recorded.

Below we describe how we calculated the accuracy and the details of the speech recognizer we used.

Calculating Accuracy Rate At this point in our iterative design we only calculated the accuracy of the boolean half of our NMF model, which estimates which lights are part of a configuration. Each of the tuples held out is divided into two parts, 1) the command (x'_{cmd}); and 2) the lighting scene (x'_{config_truth}). Let the estimated lighting scene change be x'_{config} (we use the steps described in the training section and shown in Figure 5.6 to calculate x'_{config}). x'_{config} is a vector of real values between zero and one. We want to compare x'_{config} with

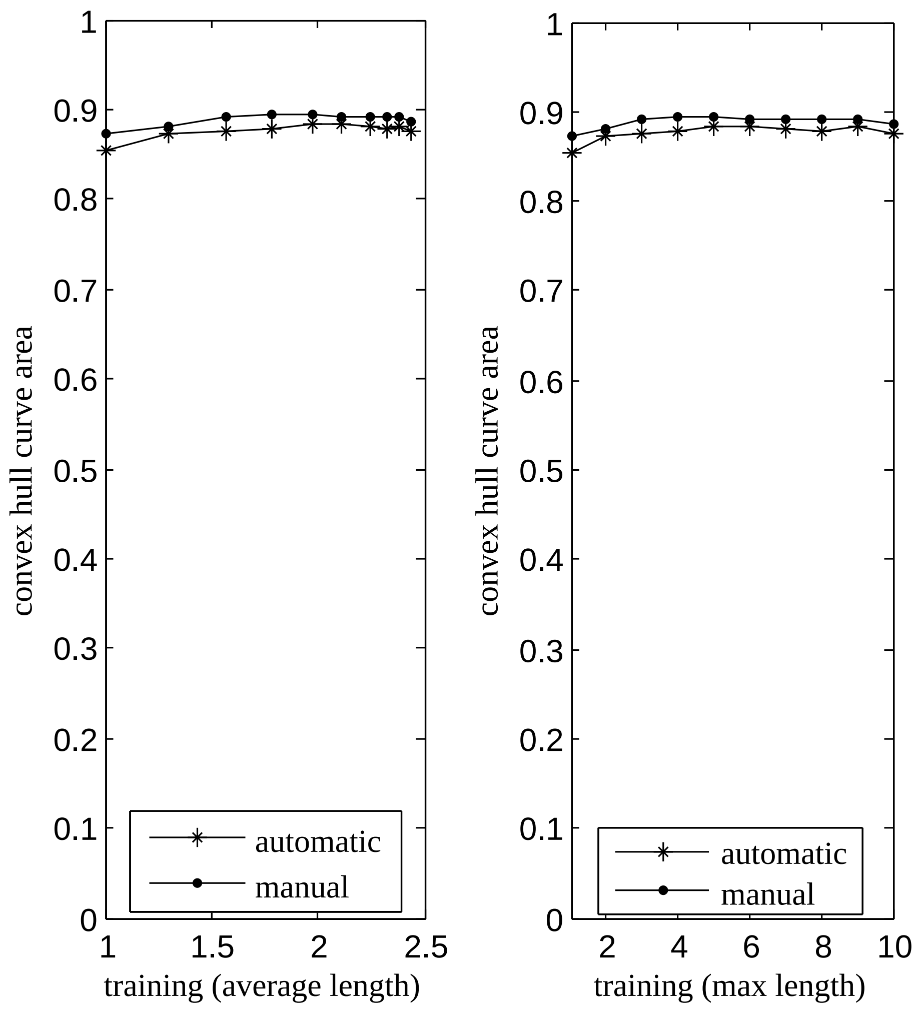


Figure 6.5: Cross validation accuracy rate as a function of training length for data with manual and automatic speech recognition transcripts. Lef: accuracy as a function of average training length. Right: accuracy as a function of max training length.

the lighting scene demonstrated by the user, x'_{config_truth} but x'_{config_truth} is a vector of boolean values. In order to compare x'_{config} and x'_{config_truth} we need to apply a threshold to x'_{config} to convert it to a vector of boolean values. A ROC (Receiver Operating Characteristics) graph [Fawcett 2003] will allow us to visualize the performance for all thresholds. The false positive rate is plotted against the true positive rate. Figure 6.6 shows the ROC graph for a model trained on a maximum of five commands per group and an average of 1.8 commands per group. The diagonal line $y = x$ in a ROC graph represents the strategy of randomly guessing which lights should be in the configuration. The area under the curve is equivalent to the probability that our model will rank a randomly chosen positive instance higher than a randomly chosen negative instance. If the curve has a dip in it, there is a range of threshold values that are less than optimal. In practice we would never select one of those thresholds, so to compare ROC graph areas, we compare the areas under the convex hull curve [Fawcett 2003] (the dashed line in Figure 6.6).

Automatic Speech Recognition Transcription In addition to calculating the accuracy with manual transcripts of the commands, we also calculated the cross validation accuracy with transcripts generated with an automatic speech recognizer on users' actual utterances.

We chose to use Carnegie Mellon's Sphinx 3 recognizer. Sphinx 3 is a state-of-the-art fully-continuous acoustic model recognizer, which is open for experimental use. Sphinx has many useful features for advanced applications: output of a full recognition lattice (not just the most likely utterance), use of class-based and dynamic language models, and speaker adaptation. Sphinx 3 supports training and plug-in of the key recognizer components, which

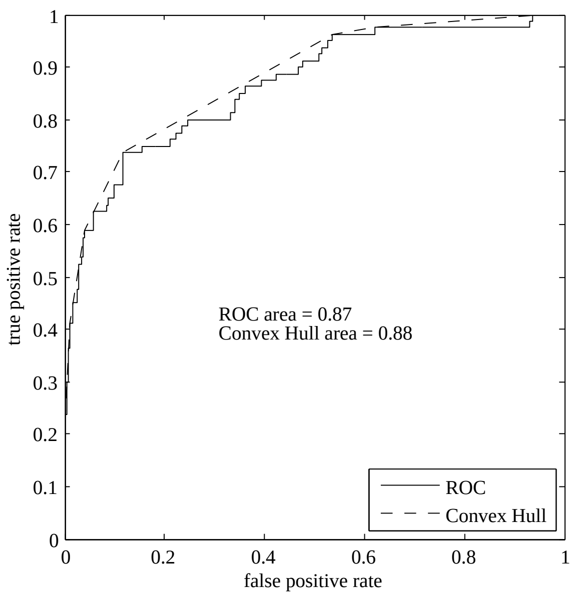


Figure 6.6: ROC graph for a model trained on a maximum of five commands per group, and an average of 1.8 commands per group.

are the acoustic model and the language model.

We felt that the publicly available acoustic models were a poor match to our user base (HUB4, WSJ, RM1 feature artificially clean speech being read from transcripts). We were able to obtain a large corpus (56 hours) of meeting transcript speech from the ICSI (International Computer Science Institute) in Berkeley. This data is direct recordings of meetings between ICSI researchers. It 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 trained several model configurations. Most accurate results were obtained with an acoustic model with 2000 senones (context-dependent sub-phonetic units), each of which was a mixture of 32 gaussians. This model did much better than the public domain models (RM1, HUB4, and WSJ) that we tried. It also had more complex senones, 32 gaussians vs. 8 for the other models. We speculate that this matches the phonetic diversity in the training and test speakers, which is much higher in our case.

Available language models were also a poor match for the lighting control speech data. We used the CMU language modeling toolkit to build a trigram language model from the earlier text data that we gathered during our first data collection phase. There were 157 utterances in this set. This is a very small set of data to be used for language modeling, but nevertheless gave acceptable results. From this model, we measured the perplexity of the actual speech transcripts recorded later and found that it was just 22. Perplexity is a measure of how well a model fits the speech data, and allows the recognizer to anticipate word combinations. The value of 22 is very low (which is good) for natural speech. Given its small source dataset, improvements and extensions to the language model are likely to cause significant improvements in recognition accuracy.

The recognizer had no speaker-specific model adaptation, nor was speaker identity available at recognition time. Because of this, we obtained better performance with the offline recognizer sphinx3_decode than the online recognizer sphinx3_livedecode. The offline decoder uses an entire utterance to do CMN (Cepstral Mean Normalization) which is highly speaker-dependent. The live decoder must do CMN in real-time with available speech data up to that point, which is invariably less accurate unless the system can use the cepstral means from previous utterances by that speaker. That was not possible in our implementation at this time. Even though shinx3_decode requires the entire utterance before recognition begins, it can still be used for interactive interfaces. The user must deal with a longer delay because recognition only starts when the utterance finishes. With sphinx3_livedecode, recognition results are available very soon after the utterance is finished.

Sphinx 3 is a highly tunable recognizer allowing speed to be traded for accuracy. We tuned it for slightly slower than real-time performance: 1.2x real-time on a 2.8 GHz machine. We were able to do this with almost no loss of accuracy. The WER (Word Error Rate) for the actual speech data was 30%, which is probably misleading. We observed that the distribution of errors is "long-tailed". That is, many utterances had no errors at all, and there were long passages of speech with error rates below 20%. But users occasionally volunteered information well outside the scope of lighting control: "Don needs light in the tool shop to find his spool of wire," which added a half-dozen word errors. Since most utterances are just a few words long, these "garden path" sentences have a big impact. Fortunately as the previous sections show, speech errors had only a moderate effect on overall performance of the interface. We suspect this is because word errors are much more likely on infrequent and non-sequitur speech (which will score low probability from the language model), but these are not salient to the NMF model. Improvements in recognizer accuracy should certainly be possible using a more comprehensive language model. But performance appears usable as is.

Qualitative Analysis

After the data collection we interviewed the participants about their experience with "training" the system on their lighting configurations and commands. Many participants mentioned their commands ended up being like a personalized light switch for their lights in the cubicle area and for the permanent public areas, such as the kitchen area and the couch. For each participant, this added up to one to three personalized light switches.

Some participants used commands that referred to other participant's lighting configurations, usually to turn off lights left on by other participants. Many participants expressed difficulty figuring out what to call the lights in the middle of the public space (far away from permanent furniture) and figuring out the mapping between the physical lights and the lights on the GUI. To address this difficulty, they requested location sensitive speech input, or a multimodal gesture and speech interface. They wanted to be able to say "more light here" or to be able to point at a group of lights and say "turn on." Both are possible, but they require more infrastructure. Location sensitive speech input requires either a large room-size array microphone or some other support for location tracking plus audio input from anywhere in the room. A multimodal gesture and speech solution is presented in Wilson's XWand work [Wilson and Shafer 2003], which uses a special wand and at least two calibrated cameras in the space for location tracking and pointing direction input. This system only supports predetermined commands with a simple command and control style grammar.

Types of Lighting Configurations In our interviews, we found the participants fell into two distinct groups with respect to when they made changes to the lighting scene. Some participants turn the lights on when they come into a space or in preparation to go into a space, while other users first evaluate the lighting conditions in a space and then may decide to change the lighting scene.

For example, one participant religiously turned on her lights before she entered her cubicle (at the public machine), and then turn her lights off again when she left the lab (as long as one of her cubicle mates were not using the shared lights). Another participant would turn on the lights over the couch before moving over to the couch to read and then turn them off again once she was done using the couch area.

The users who did not turn their lights on as soon as they moved into a space talked about evaluating the ambient lighting before deciding whether to make a change to the lighting scene around them. At this point, they did not want to have to get up from where they were to change the lighting scene, so having the ability to change the lighting scene from many places in the lab is very important to them.

Graphical User Interface When asked about their experience using the web based graphical user interface, some users said a more polished GUI would be fine when at their desks. Other participants didn't like using the GUI to control the lights because their computer was not necessarily on and getting up to use the public machine was too much of a hassle.

Drawbacks of Speech Input Two users expressed discomfort using speech in the workspace to control the lighting scene. They said it drew attention to themselves and made them feel uncomfortable.

Advantages of Speech Most participants commented how a speech interface would allow them to have a customized dimmer for the sets of lights they commonly use. A few participants also commented that speech is a good option when your hands are full or dirty and you want to change the lighting scene.

Training When asked to speculate as to how long they would be willing to train the system, the answers were all over the board. Some participants said they would be willing to train the system for up to a month, while others said they would be willing to demonstrate their desired lighting scene once or twice, and then they would be willing to correct each kind of error once. Some said it would depend on the situation, if they were just running in and out quickly or were very busy, they would not want to record a training data point, but if they weren't in a hurry it wouldn't be a problem.

In practice some participants were very regular in their commands right from the start, while others took a few times to settle on a command. Some had a few variations of the same command they alternated between.

Overall we got positive feedback on the possibility of a speech interface for the workspace lighting control application, but no interface is ideal for every user or every situation.

6.4.3 Discussion

This study guided the design of both the qualitative and quantitative aspects of the system. We got a better understanding of the users' experiences and opinions about using speech to control their lighting configurations. Some users found speech to be a good fit for them while others preferred to use a different interface. Any complete solution for a group of users should provide a variety of interfaces so that all users can find an interface that works well for them given a particular task. One user may even use a different interface for different tasks. We got more evidence from this study, which was further confirmed in our second deployment, that a user-extensible speech interface is a good fit for some of the users, making it an important interface to consider including in a complete system.

The high-fidelity formative training data once again demonstrated both the configurations and commands are overlapping, which strongly suggests the selection of NMF as the basis for the learning algorithm. This was also reinforced by the promising accuracy results. The training data directly informed the design of the learning algorithm, which in turn enabled us to build and deploy a live system. In the next section we present the study we ran with the first version of the live Illuminac system. That study allowed us to measure the accuracy from the users' perspective.

6.5 First Live Deployment

In the previous section, we presented some initial accuracy measurements based on the formative training data and an early implementation of the learning algorithm. These results do not take into account users' tolerance level for false positives or false negatives. In this section we present a study where we looked at the users' experience with a live deployment of Illuminac.

6.5.1 Study Description

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. A training mode was added to the first version of the GUI to allow users to record their training points (shown in Figure 6.7). To add a training point, users were asked to complete the following four steps:

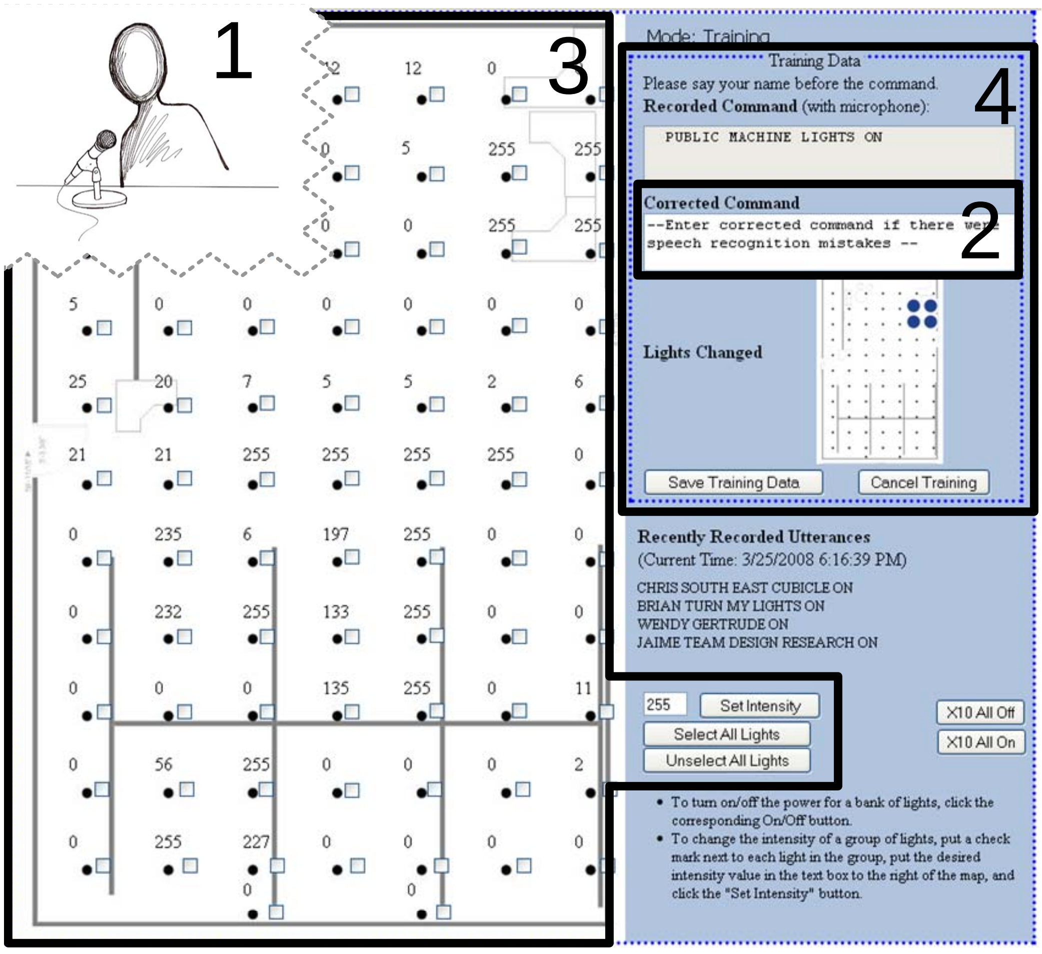
1. record her command using one of the microphones around the room (prefaced by her name to simulate speaker identification);
2. correct any recognition errors in the automatic speech recognition transcript using the web-based graphical user interface;
3. demonstrate her desired lighting scene using the web-based graphical user interface to manually set the lighting configuration; and
4. verify the command and configuration are correct before saving the training point.

We started with no training data and asked the participants to train the system on their commands again[1]. 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	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. The questionnaire is included in Appendix A.

[1]Many of the participants in this study also participated in previous studies and had already provided training points.



Please say your name before the command.
Recorded Command (with microphone):

```
PUBLIC MACHINE LIGHTS ON
```

Corrected Command

```
--Enter corrected command if there were
speech recognition mistakes --
```

Recently Recorded Utterances
(Current Time: 3/25/2008 6:16:39 PM)

CHRIS SOUTH EAST CUBICLE ON
BRIAN TURN MY LIGHTS ON
WENDY GERTRUDE ON
JAIME TEAM DESIGN RESEARCH ON

- To turn on/off the power for a bank of lights, click the corresponding On/Off button.
- To change the intensity of a group of lights, put a check mark next to each light in the group, put the desired intensity value in the text box to the right of the map, and click the "Set Intensity" button.

Figure 6.7: The first version of the training interface. To add a training point, users complete the following four steps: (1) record her command using one of the microphones around the room (prefaced by her name to simulate speaker identification); (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; and (4) verify the command and configuration are correct before saving the training point. (The drawing of the user recording her command is by Lora Oehlberg.)

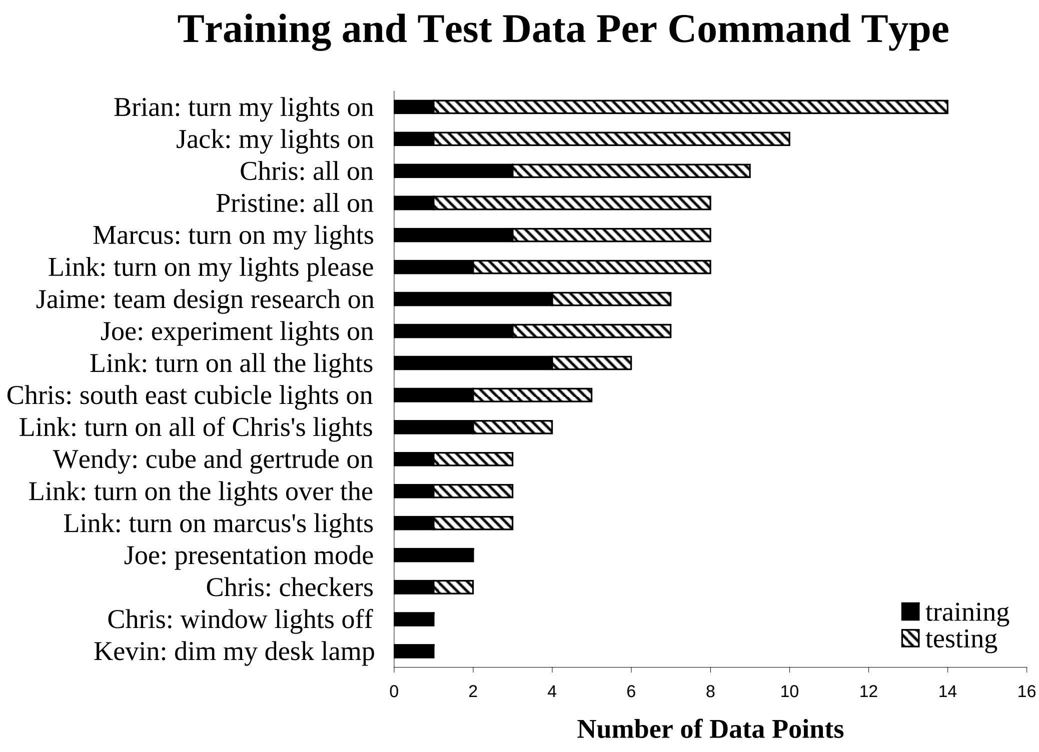


Figure 6.8: The number of training and test data points collected for each group of similar commands. The command group labels are representative commands recorded by the participants.

6.5.2 Data Collected

We collected 43 training points and 81 test points from the 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 that the participant provided using one of the microphones, and the participant's evaluation of the system's response. Figure 6.8 shows the number of data points collected for each group of similar commands.

6.5.3 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 6.8 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 6.9). 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.

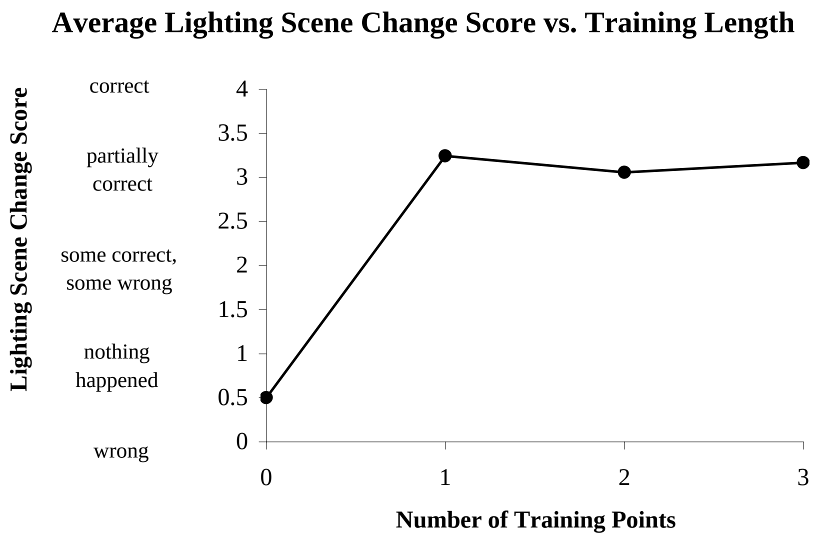


Figure 6.9: The average test point score as reported by the participants versus the training length for the specific type of command when the test point was recorded

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*. One of the participants who responded *Maybe* said the microphone was too far away, and he was too lazy to get to a microphone, which is a limitation in our experimental setup that could be overcome in a commercial deployment. Right now each cubicle with three people shares one microphone, which 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 does not often use the same set of lights. In such an open space, a location-based speech approach would likely work better where users could say “lights on here.” Such a location based approach could be implemented with distributed microphone array technology overhead, though such technology would not be desirable in the cubicle area for privacy reasons. With overhead microphones, users cannot control what is being 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 at least similar to the performance during the study. We corrected this problem in the next version of the training interface, which we used in the second live deployment (Section 6.6).

6.6 Second Live Deployment with Graphical User Interface Control

6.6.1 Methodology

We propose a user-extensible speech interface to address the challenge of controlling the ever increasing number of devices in our environment. The interface should allow users to use commands that are natural to them and are easy to remember. These commands should apply configurations which are customized to the space and its users. In order to support such customization, users invest a small amount of time training the system. These requirements raise many questions about how such a solution works in practice. Can users remember their commands? How many different configurations do they use in practice? How many training points do they have to provide? When is the system a good fit and when is another kind of interface necessary? We explore these questions through a twelve week deployment of Illuminac. Below we describe the methodology behind our study.

Participant Pool

The participants in the study are all regular occupants of the space. This property is important to be able to ensure the configurations they train and use the system on are naturally occurring configurations.

Study Style

We ran an *in situ* study to get a better understanding of how the system fits into or does not fit into regular usage of the space and the lights. It also allows the configurations and commands to be used in a natural way, rather than an artificial way such as in a batch laboratory study.

Control Case

We used a web-based graphical user interface as the control case (shown in Figure 5.5). It has a similar deployment cost as a user-extensible speech interface. Both kinds of interfaces have to be customized to work with the devices in the space but neither have to be customized to work with the specific configurations used in the space. Note that Illuminac is customized to work with the devices in the space at the time of deployment, but the customization to work with the specific configurations is done by the users.

We considered including a hardware interface as a control case, but such an interface is considerably more costly to customize to a new set of devices and does not scale as the number of devices grows as a speech interface and a graphical interface do.

Design of Interfaces

Both Illuminac and the graphical user interface were designed, implemented, and iterated on separately. This iterative development of both interfaces is important to ensure we are comparing Illuminac against a well designed and usable control case. The graphical user interface had been deployed and in use for two years. The version used in the study was the third major version with changes made for each new version based on feedback from daily users.

Illuminac was designed and developed with an iterative design cycle including a lo-fidelity text phase to understand the way occupants use the lights, a high- fidelity training data collection phase, a formal week long deployment to evaluate, as well as an informal semester long deployment where any occupants in the lab who wished to use the system could do so and give informal feedback.

6.6.2 Study Details

We ran a within-subjects study over twelve weeks with eight participants. Each of the participants were able to participate for a different length of time with an average participation time of 7.9 weeks, minimum participation time of four weeks, and longest participation time of eleven weeks.

Since the study is an *in situ* study, the participants were asked to use one of the interfaces each time they wanted to change the lighting scene in the workspace. They were asked not to use the light switches that turn on all the lights but rather to use one of the interfaces that give move flexible control over the lighting.

Phases: GUI, Illuminac, and Mixed

The study was divided into two exclusive phases: the GUI phase and the Illuminac phase followed by a mixed phase. To mitigate order effects, the order of the GUI and Illuminac phases was randomized with half of the participants starting with the GUI phase followed by the Illuminac phase and the other half of the participants doing the phases in the opposite order (the structure of the study is summarized in Table 6.1). During the GUI phase the participants were asked to use the graphical user interface exclusively. During the Illuminac phase the participants were asked to use the Illuminac system as much as possible but were allowed to use the graphical interface if the Illuminac system did not work for them. After the exclusive phases, the participants took part in the mixed phase where they were given the option to use either interface. They were asked to make a choice each time they wanted to change the lighting scene. That is, they did not have to select one interface or the other for the whole phase but could swap back and forth between interfaces depending on which one was more appropriate for the current task. At the beginning of each phase, the participants were given a short tutorial on how to use the interface for the given phase, and participants were free to ask clarification questions about the interfaces throughout the study. Each phase lasted approximately one third of the duration of the user's total participation time.

	Phase I	Phase II	Phase II
Group A	GUI	Illuminac	Mixed
Group B	Illuminac	GUI	Mixed

Table 6.1: The study contained three phases: GUI–participants used the GUI exclusively; Illuminac–participants used Illuminac exclusively; Mixed–participants used GUI and Illuminac as they saw fit.

Questionnaires and Exit Interview

Each participant was asked to complete a short questionnaire after each of the GUI and Illuminac phases (see Appendix B) asking about their experience with the interface from the recently completed phase.

We concluded the study with an exit interview after the mixed phase. Before each interview, we reviewed the questionnaires about the participant's experience with the GUI and Illuminac interfaces and reviewed their usage data. This helped guide the interview to include questions about specific practices in addition to overall feedback. Participants were asked how they decided when to use the GUI and when to use the Illuminac system. If they had technical difficulties with one of the interfaces, we asked them to hypothesize about their practices had the technical difficulties been resolved.

Description of Web-Based Graphical Interface

The graphical user interface (shown in Figure 5.5) we used as the control case is a web-based interface implemented in Flash. The graphical interface was always open on a public machine at the entrance to the room. In addition to the public machine, all of the participants had access to the interface on their personal computers.

The interface uses a painting metaphor based on feedback from several users. Since each light has two controls—intensity and power—both controls are accessible through the interface. The user selects an intensity or a power setting from the pallet on the right hand side of a floor plan representation of the lights and paints on the desired configuration. A configuration can consist of many different intensities or power settings. The user then clicks on the "Apply" button.

6.6.3 Results

Control Case

We found the graphical user interface was easy to use and easy to learn how to use and thus a fair control case. The average for ease of use was 4.4 out of 5 and the average for how easy the interface was to learn was also 4.4 out of 5 on the following scale for the following two questions:

Using the GUI to change the lighting scene in the lab is:
Learning to use the GUI was:

1	2	3	4	5
Very Hard	Hard		Easy	Very Easy

Mixed Phase

When given the choice to use the Illuminac system or the graphical user interface to change their lighting scene, there was a big variation in usage patterns. Two participants clearly preferred Illuminac over the graphical user interface, two participants used the graphical user interface slightly more than the Illuminac system and four participants used the graphical user interface almost exclusively if not exclusively (see Figure 6.10).

Based on the results from the exit interviews and the questionnaires, three participants had little trouble with the system, used it regularly, and found it quick and easy to use. Three participants said when the system worked, it was a good way to control the lights, and if the errors were significantly reduced, they would use the system. Finally two participants preferred to use another interface for all of their lighting control tasks. We discuss these results in more detail below.

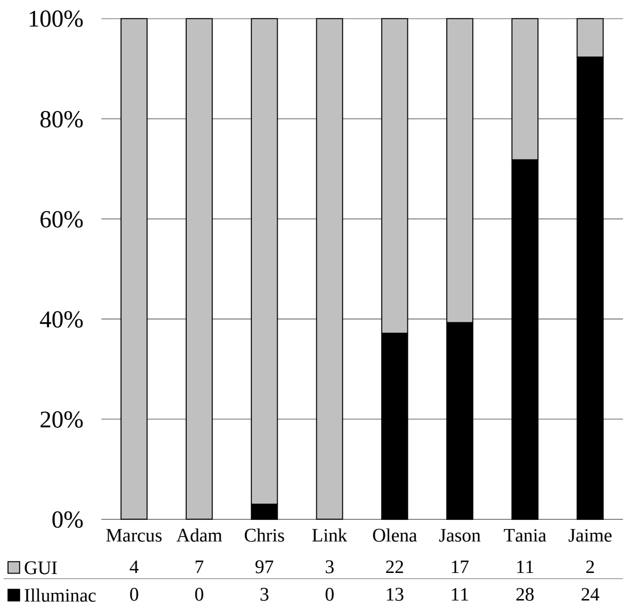
Illuminac Success Stories Tania, Jaime, and Jason all found the Illuminac system a good way to control the lights. In their post Illuminac phase questionnaires, they said:

> *It worked! It was really easy to learn to train, and I hardly had to train it too much. It's been good enough that I can even change my phrases some times and it actually recognizes well. It responds quickly to my commands. I really like it.*

> *It's VERY easy for me to just lean over and adjust my lights from my desk, without having to open up a separate window on my computer. I prefer knowing that it's there so that regardless of whether or not I have a GUI open, I can turn on my lights.*

> *It is rather novel. For the task of turning on and off a specific set of lights (...) or adjusting their brightness to one of the three levels I trained, using the voice commands was quick and easy. Also, I did not need access to a computer or specific bank of light switches to turn on just the lights that I needed; hence, I could tell the lights to turn on and it would chug away and turn on the lights while I was setting up my computer.*

Mixed Phase: Percentage Illuminac and GUI Used



	Marcus	Adam	Chris	Link	Olena	Jason	Tania	Jaime
▨ GUI	4	7	97	3	22	17	11	2
■ Illuminac	0	0	3	0	13	11	28	24

Participants' Experience with Illuminac Based on Questionnaires and Exit Interviews

	Marcus	Adam	Chris	Link	Olena	Jason	Tania	Jaime
Prefer Illuminac							●	●
Like using both Illuminac and GUI			●	●	●	●		
Prefer GUI	○	○						
Poor speech recognition			⊙	⊙				
Trouble training Illuminac	⊙		⊙		⊙			

Promising ● Fixable ⊙ Non Fixable ○

Figure 6.10: *Top:* Quantitative results from the mixed phase. The percentage of times each participant used Illuminac versus the GUI. *Bottom:* A high level description of the participants experience with Illuminac based on the questionnaires and exit interviews.

Jason ended up using the GUI more than the Illuminac system in the mixed phase because the GUI allowed him to turn off the power for his lights where as the Illuminac system just sets the intensity to zero but leaves the power on. This limitation is an artifact of the hardware available when we installed the individually controllable lights. In his exit interview, he said had the power also been turned off with the Illuminac system he would also have used it to turn off his lights in addition to turning his lights on.

Poor Speech Recognition Two of the participants experienced particularly poor speech recognition results that severely interfered with their ability to use the Illuminac system. The large discrepancy in speech recognition results suggests the speech recognition accuracy for these users could benefit greatly from speaker adaptation in the speech recognizer. This feature is supported in our speech recognizer (Sphinx 3.5 [Placeway et al. 1997]) and could be incorporated into the Illuminac system.

Trouble With Training Three of the participants experienced trouble training the system. This was a result of a bug we found in the system half way through the study that made it so that when new training points were added, they drowned out the older training points. Although we fixed this bug, Olena had already lost confidence in the system and would just use the graphical interface because she assumed the Illiminac system would not recognizer her commands. In her exit interview and in her Illuminac phase post questionnaire she explained she liked the system and would likely use it if the system did not “forget” her commands when other participants added their commands.

Preference for Alternate Interface Marcus and Adam preferred not to use the Illuminac system. Marcus only wanted to have to use one interface. He did not want to have to decide which interface to use based on his lighting control task. He found it much easier to switch to a different window and click a few times to change his lighting scene than leaning over to speak his command into the microphone at his desk. Adam preferred the light switch on the wall over the GUI and the speech interface. This is the switch that turns on all the lights on together. If he was required to use the individually controllable lights, he preferred to use the GUI. He spends most of his time working in the public space, often under a different set of lights and had trouble remembering what he named the different lighting configurations in the space.

Commands and Configurations

The number of configurations used by each participant ranged from one to eight with an average of 3.1 per participant. The number of training points provided for each configuration ranged from 1 to 8 with an average of 2.5 per configuration. Table 6.2 lists representative commands for each configuration and the number of training points per configuration.

The participants who provided high numbers of training points (Marcus, Chris, and Olena) were the participants who had trouble training the system as described above. Had we made the corrections listed above these participants would not have needed to provide as many training points.

User	Command	Training Points
Marcus	*marcus turn my lights on*	6
	marcus turn on public computer lights	1
	marcus turn all lights off	1
Adam	*adam center lights on*	1
Chris	*chris everything on*	5
	chris south east cubicle lights on	3
	chris common area lights on	2
	chris public area lights on	2
	chris my desk lights on low	2
	chris my desk lights on medium	2
	chris center of public area on	2
	chris marcus's lights on	2
Link	*link turn my cube lights on*	3
	link all lights on	2
	link turn my desk lamp on	2
Olena	*olena desk lights on*	2
	olena ME cube lights on	2
	olena cube light on	8
Jason	*jason desk lights to full*	1
	jason desk lights to half	1
	jason desk lights to dim	1
Tania	*tania turn on my lights*	3
Jaime	*jaime team design research on*	4
	jaime ethel and gertrude on	2
	jaime marcus is a rock star	3

Table 6.2: Representative command for each lighting configurations users trained the system on and the number of training points provided for each configuration. Note: users were asked to state their name at the beginning of the command to simulate speaker identification.

Remembering Commands For the most part the participants did not have trouble remembering their commands. None of the participants trained one configuration with multiple commands and most of the configurations were used more than once after they were trained. For example, Tania was able to remember her command even after a six week trip, and Jason was able to remember his commands after three weeks including a one week trip and the two weeks when he was in the GUI phase of the study. Three participants (Marcus, Adam, and Chris) mentioned they had trouble remembering their commands in the large public space where the furniture tends to move around on a daily basis. They had trouble remembering their commands because they had trouble coming up with a name for the configuration to begin with. In places where there was a fixed point of reference the participants were able to come up with names for their configurations much more easily and as such were able to remember them more easily.

Selecting an Interface There are a number of common factors involved in deciding which interface to use when changing the lighting scene. They include the time relative to the participant's stay in lab, whether they were comfortable using voice commands with others around, the frequency of use of the configuration, and the type of change being made to the lighting scene.

Time within Stay The changes to the lighting scene can be broken down into three groups based on when they are performed relative to when the participant arrived in lab and when they left: those performed as the participant arrived in lab, was working in lab, and left the lab. One of the main advantages of the Illuminac system is being able to change the lighting scene while performing another task. Some participants liked using the speech interface as they came into lab to turn on their lights while they were unpacking their laptop or logging into their desktop computer and setting up. This allowed them to turn on the lights before having set up their computer. Others liked to use the speech interface as they were packing up at the end of the day, once again in parallel with other activities.

In the middle of the day some participants liked not having to switch to a new window with the GUI open to change the lighting scene but could instead lean over and speak their command into the microphone near their desk. However, other participants found it easier to switch to another window to control the lights with the GUI than having to lean away from their computer toward a microphone to speak their command.

Use of Spoken Command One of the potential drawbacks of a speech interface is having to make noise to use the interface. Some participants were self conscious about using a voice command when others were around while others did not mind at all. One participant preferred not to use voice commands in the middle of his stay, but did not mind using them as he was leaving even if there were other people around because it served as a way to say goodbye to those around him.

Configuration Frequency of Use The participants who tended to use the open public space more often preferred to use the graphical interface because the frequency with which they used the configurations in that space was much lower. This low frequency of use in addition to the lack of permanent furniture also made it hard to come up with names for the configurations and as a result, the configurations in the space were harder to remember.

Type of Lighting Scene Change There are two kinds of lighting scene changes: absolute and relative. For example an absolute command might be *turn my desk lights to dim.* A relative command might be *turn my desk lights up.* Currently the Illuminac system only supports absolute changes and as a result, the participants expressed a preference toward the graphical interface when they wanted to make a relative change to the lighting scene. The Illuminac system can theoretically support both, we just selected the one used more commonly based on our initial study of usage patterns in the lab to begin with.

Central Versus Local Access

In addition to studying the advantages and disadvantages of the Illuminac system and the graphical interface, we also got feedback about where users wanted to have access to the lighting controls. The participants had access to both systems at a central location as well as in their cubicles. Once again the usage patterns varied widely among the participants. Some participants always used the centrally located controls independent of which interface they were using. They were used to going to a bank of light switches and treated the centrally located controls as more sophisticated light switches. Other participants really liked being able to control the lighting scene without having to get up from their desks. These participants tended to make more adjustments to the lighting scene in the middle of their stay. And other participants liked having both options. One participant liked to use the centrally located controls as he came in and the controls at his desk as he left. Another participant liked to use the controls in the opposite manner.

6.6.4 Discussion

The study results support the hypothesis that a user-extensible speech interface is an important component of a complete solution to controlling a large number of devices in an environment. Illuminac gave users a quick and easy way to control the lighting scene for frequently used configurations, especially when the users wanted to control the lights while doing another task and was a good compliment to the GUI, which worked better for infrequently used configurations. Many participants were willing to train the system on their customized commands and liked using Illuminac as long as they did not need to provide more than one or two training points, which is possible with the current system. In summary, users were willing to train the system, able to remember their commands, and willing to use voice commands to control the lighting in their workspace in many cases.

As mentioned above, for workspace lighting control, users are more likely to adjust their lighting if they have access to an interface that is convenient and easy to use. When users adjust their lighting, they often use less energy than if the lights were set to a default value. With diverse interface preferences and usage patterns, it is important to include a diverse set of interfaces. A user-extensible speech interface would compliment a complete solution including, for instance, an intelligent daylighting system, and a manual graphical or hardware interface.

We believe an easy to use interface is just as important in task-specific device control in other environments such as the home and our results should transfer to those domains. That is, some types of users will find a user-extensible speech interface useful and easy to use in those environments, they will also be willing to train the system and be able to remember their commands.

Study Limitations

Since all of our study participants were engineering students we can only hypothesize about other types of users. The Illuminac system could have been at a disadvantage among users who are very comfortable using graphical user interfaces and who spend a significant amount of time in front of a computer where the GUI lighting control was easily accessible. As long as they are comfortable training the system, some less computer savvy users many find a user-extensible speech interface to be a better fit.

Our study is also limited by the domain in which it was conducted, the workplace. In the workspace it was reasonable to have a dedicated computer for the lighting control GUI and computers through which the GUI could be easily accessed were prevalent throughout the space. In a home environment, distributed microphones or the use of telephones to control the environmental state may be more appropriate. They would not stick out as much as a computer screen, mouse and keyboard in a space such as the living room or kitchen. This might give a user-extensible speech interface more of an advantage in the home environment.

Chapter 7

Conclusion

The number of electronic devices in our environment is every increasing. While this brings greater flexibility and control, configuring each individual device becomes ever more tedious.

I have argued an interface for controlling environments with ever increasing numbers of electronic devices should be calm, easy to learn and use, support control of configurations of devices, and allow rapid definition and recall of favorite configurations. Supporting control of configurations of devices allows the user to control and environment such as the home or workplace on a task or activity level instead of on a individual device level. This brings the interaction closer to the user's mental model. Allowing rapid definition and recall of favorite configurations ensures the interface supports the configurations used in practice, even as the configurations change over time. This also means an interface designed for one space can be transfered to another space within the same sub-domain.

We propose a user-extensible speech interface for controlling this type of ubiquitous computing environment. The novel aspect is that the system learns configurations and names for those configurations simultaneously to support the specific configurations used by the users and names for those configurations most natural and easy to remember for the users.

I presented a specific realization of our proposed solution to controlling configurations of devices in ubiquitous computing environments, Illuminac, through which we studied and evaluated our proposed solution. Illuminac is a user-extensible speech interface for workspace lighting control. The workspace for which Illuminac is designed and in which it is deployed has 79 devices (individually controllable lights) that are controlled by 25 users who have both their own and shared environment names and configurations. In summary I make the following contributions:

- I identify and describe an approach to the Simultaneous Naming and Configuration problem (SNAC) — a problem that arises in natural environment control. Our approach results in a calm interface that learns intuitive *names* for commonly used *configurations* of devices in a home or work environment instead of requiring the user to learn hard coded, static names for hard coded configurations of devices, or worse, hard coded names for individual devices.

- I identify an appropriate learning algorithm for the simultaneous naming and configuration problem: non-negative matrix factorization (NMF).

- I 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, I 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.

- I show users are willing and able to train the system to work for their favorite configurations, able to remember their commands, and prefer to use voice commands over using a graphical user interface to control the lighting scene in many cases. I believe these results should transfer to other task-specific control of a collection of devices in other environments such as the home.

Bibliography

Adura Technologies. URL http://www.aduratech.com.

Kobus Barnard, Pinar Duygulu, Nando de Freitas, David Forsyth, David Blei, and Michael I. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.

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

Dan Bohus. Roomline: a spoken dialog system that provides assistance for conference room reservation and scheduling. URL http://www.ravenclaw-olympus.org/roomline.html.

Richard A. Bolt. "put-that-there": Voice and gesture at the graphics interface. In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, volume 14, pages 262–270. ACM Press, July 1980.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick

Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Comput. Linguist.*, 16(2):79–85, June 1990.

John Canny. GAP: a factor model for discrete data. In *Special Interest Group on Information Retrieval (SIGIR)*, pages 122–129. ACM Press, 2004.

Nils Dahlbäck and Arne Jönsson. Wizard of Oz studies – why and how. In *Proceedings of Intelligent User Interfaces (IUI)*, pages 193–200, 1993.

DALI. Digital addressable lighting interface (dali). URL `http://en.wikipedia.org/wiki/Digital\discretionary{-}{}{}_Addressable\discretionary{-}{}{}_Lighting\discretionary{-}{}{}_Interface`.

S. Escuyer and M. Fontoynont. Lighting controls: a field study of office workers' reactions. *Lighting Research and Technology*, 33(2):77–94, June 2001.

Tom Fawcett. Roc graphs: Notes and practical considerations for researchers. Technical Report HPL-2003-4, HP Labs, 2003.

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. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Joshua Juster and Deb Roy. Elvis: situated speech and gesture understanding for a robotic chandelier. In *International Conference on Multimodal Interfaces (ICMI)*, pages 90–96, 2004.

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

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

James Love. Field performance of daylighting systems with photoelectric controls. In *Proceedings of Right Light Three, 3rd European Conference on Energy Efficient Lighting.*, pages 75–82, 1995.

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

T. Moore, D. J. Carter, and A. I. Slater. A field study of occupant controlled lighting in offices. *Lighting Research and Technology*, 34(3):191–202, September 2002.

Michael 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.

Nuance. Formerly bevocal, a leading provider of hosted application systems for customer self-service. URL `http://www.bevocal.com`.

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.

Jose F. Quesada, Federico Garcia, Ester Sena, Jose A. Bernal, and Gabriel 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.

Manny Rayner, Pierrette Bouillon, Beth A. Hockey, Nikos Chatzichrisafis, and Marianne Starlander. Comparing rule-based and statistical approaches to speech understanding in a limited domain speech translation system. In *International Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, 2004.

Vipul Singhvi, Andreas Krause, Carlos Guestrin, James H. Garrett, and Scott H. Matthews. Intelligent light control using sensor networks. In *Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSys)*, pages 218–229, New York, NY, USA, 2005. ACM.

TellMe Networks. Fundamentally improving how people and business use the phone. URL `http://www.tellme.com`.

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

Yao-Jung Wen, Jessica Granderson, and Alice M. Agogino. Towards embedded wireless-networked intelligent daylighting systems for commercial buildings. In *SUTC '06: Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing - Vol 1 (SUTC'06)*, pages 326–331, Washington, DC, USA, 2006. IEEE Computer Society.

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

Appendix A

First Live Deployment Post Questionnaire

1. Adding a training data point (recording a command and demonstrating a lighting scene)

●	●	●	●	●
Takes Too Much Time				Is Quick and Easy

2. How many training data points would you willing to provide to be able to use speech to control the lights?

3. After the study is over, would you like to continue using the system?

   Yes Maybe No

4. Why or why not? (Would you like to continue using the system)

5. Comments / feedback on the speech interface.

6. Comments / feedback on the graphical user interface.

Appendix B

Second Live Deployment Questionnaires

B.1 GUI Phase Post Questionnaire

1. Using the BiD Lights GUI to change the lighting scene in the BiD lab is:

1	2	3	4	5
Very Hard	Hard		Easy	Very Easy

2. Learning to use the BiD Lights GUI was:

1	2	3	4	5
Very Hard	Hard		Easy	Very Easy

3. What did you like about the BiD Lights GUI?

4. What didn't you like about the BiD Lights GUI?

5. What did you find confusing about the BiD Lights GUI, if anything?
6. What was clear / intuitive about the BiD Lights GUI?
7. What would you change about the BiD Lights GUI?
8. In what situations did you find the BiD Lights GUI a good way to change the lighting scene?
9. Were there situations in which you wish you had a different way to control the lights?
10. Is there anything else you would like to add?

B.2 Illuminac Phase Post Questionnaire

1. Using the BiD Lights Speech Interface to change the lighting scene in the BiD lab is:

1	2	3	4	5
Very Hard	Hard		Easy	Very Easy

2. Learning to use the BiD Lights Speech Interface was:

1	2	3	4	5
Very Hard	Hard		Easy	Very Easy

3. Training the BiD Lights Speech Interface was:

1	2	3	4	5
Very Hard	Hard		Easy	Very Easy

4. Learning how to train the BiD Lights Speech Interface was:

1	2	3	4	5
Very Hard	Hard		Easy	Very Easy

5. Training the BiD Lights Speech Interface so that it worked for my commands and lighting scenes required:

1	2	3	4	5
Too Much Effort		A Reasonable Amount of Effort		Not Much Effort at All

6. What did you like about the BiD Lights Speech Interface?

7. What didn't you like about the BiD Lights Speech Interface?

8. What did you find confusing about the BiD Lights Speech Interface, if anything?

9. What was clear / intuitive about the BiD Lights Speech Interface?

10. What would you change about the BiD Lights Speech Interface?

11. In what situations did you find the BiD Lights Speech Interface a good way to change the lighting scene?

12. Were there situations in which you wish you had a different way to control the lights?

13. Is there anything else you would like to add?

Appendix C

Stop Phrase List

We used the following stop word list to exclude words from the commands that are not related to the names of configurations in the domain of workspace lighting control. It is meant to be a lower bound on the words not related, so some unrelated words will not be filtered out, but related words should not be filtered out. The stop word list was derived from the sample training data we collected during our formative, high-fidelity training data collection study described in Section 6.4.

- above
- can I
- can you
- done
- get
- I need

- I want
- in
- light
- off
- ok
- on
- out
- over
- please
- some
- thank you
- thanks
- the
- to
- turn
- up