Doppelgänger: A User Modeling System

by

Jonathan L. Orwant

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY June 1991

© 1991 Jonathan L. Orwant

The author hereby grants to MIT permission to reproduce and to distribute copies of this thesis document in whole or in part.

Signature of Author: ____________________________

Department of Electrical Engineering and Computer Science
May 20, 1991

Certified by: ________________________________

Walter Bender
Principal Research Scientist, Media Laboratory
Thesis Supervisor

Accepted by: ________________________________

Leonard A. Gould, Chairman
Department Committee on Undergraduate Theses
Doppelpäger: A User Modeling System

by

Jonathan L. Orwant

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 1991 in partial fulfillment of the
requirements for the Degree of
Bachelor of Science in Computer Science and Engineering

ABSTRACT
Computers are not personal; they do not automatically tailor their actions toward individual users. Maintenance of user models containing knowledge of user habits, interests, plans, preferences, knowledge, abilities, misconceptions, and schedules provides the computer with the knowledge it needs to perform this personalization.

Doppelpäger, a user modeling system, is described. Doppelpäger continually constructs and refines models of individuals, providing this information for use by unspecified client applications. Unlike other user modeling systems, Doppelpäger senses information about the user through many different channels, makes implicit generalizations, allows users to view and modify their models, provides an extensible architecture, interpolates between and extrapolates from models, and attempts to exploit fully the user's computational environment toward these ends. Doppelpäger's motivations, implementation, and implications are discussed in this thesis.

Thesis Advisor: Walter Bender
Principal Research Scientist, Media Laboratory

This work was supported in part by International Business Machines Incorporated.
Contents

1 Introduction ........................................... 8
  1.1 How To Read This Thesis ......................... 8
  1.2 Overview ........................................ 8
  1.3 The Need: Personalization ....................... 9
  1.4 The Solution: Doppelgänger .................... 10

2 User Modeling: The Field .......................... 13
  2.1 Overview of the Field ........................... 13
  2.2 The Scope of User Modeling ..................... 14
    2.2.1 What Constitutes a User Model? ............ 14
    2.2.2 User Modeling Dimensions ................. 15
  2.3 Research Areas in User Modeling ................. 15
    2.3.1 Student Modeling ......................... 16
    2.3.2 Plan Recognition ........................... 17
    2.3.3 Inference Mechanisms ...................... 18
    2.3.4 Dialog Systems ............................ 19
    2.3.5 Data Acquisition ........................... 22
  2.4 User Modeling in News ......................... 22
    2.4.1 Electronic Newspapers ..................... 22
    2.4.2 Information Retrieval ...................... 23

3 The Future of User Modeling ....................... 25
  3.1 Trends .......................................... 25
    3.1.1 Current Trends in User Modeling .......... 25
    3.1.2 Planning .................................. 26
  3.2 Problems with User Modeling ................... 27
3.2.1 Data Acquisition ........................................ 27
3.2.2 Planning Revisited ..................................... 28
3.2.3 Stereotypes .............................................. 28
3.2.4 Intrusiveness ............................................ 28
3.2.5 Role of User Modeling in a Larger System .......... 29
3.2.6 Architecture ............................................ 29

4 The Doppelgänger System ................................. 30
  4.1 Overview .................................................. 30
  4.2 Data Acquisition: Sensors and Observers ............ 30
  4.3 The Operator Language .................................. 35
  4.4 Doppelgänger’s Inference Engine .................... 35
  4.4.1 The Deducer and Inducer ......................... 35
  4.4.2 Chains of Reasoning ................................. 37
  4.4.3 So Where’s The Knowledge? ...................... 37
  4.5 Clients Using Doppelgänger Data .................... 37
  4.5.1 Accessing the Data .................................. 37
  4.5.2 Client Feedback, or “How Am I Doing?” .......... 38
  4.5.3 Extensible Architecture ............................ 38

5 Implementation .............................................. 40
  5.1 Context ................................................... 40
  5.1.1 Presentation .......................................... 41
  5.1.2 Transcoding .......................................... 41
  5.1.3 Personalization ....................................... 41
  5.2 Data Structures .......................................... 42
  5.3 The Sensors ............................................... 43
  5.3.1 UNIX finger ........................................... 43
  5.3.2 UNIX aliases .......................................... 43
  5.3.3 UNIX lastcomm ........................................ 44
  5.3.4 UNIX calendar ........................................ 44
  5.3.5 Badger ................................................ 44
  5.3.6 Free Text .............................................. 44
  5.3.7 Newspace .............................................. 45
5.3.8 Model Manipulation .................................................. 45
5.4 The Inference Engine .................................................. 46
  5.4.1 How Deducers Work ................................................. 46
  5.4.2 How Inducers Work ................................................ 47
5.5 Communication .......................................................... 48
  5.5.1 Operator Language .................................................. 48
  5.5.2 Process Control .................................................... 48
  5.5.3 Server-Client Communication ................................. 49
5.6 Hardware Platform ..................................................... 49

6 Issues ................................................................. 50
  6.1 Privacy .................................................................. 50
    6.1.1 Big Brother ...................................................... 50
    6.1.2 Assurances ........................................................ 51
  6.2 Doppelgänger As An Aid To Introspection .................. 52
  6.3 Models of Models .................................................... 52
  6.4 Intrusiveness and Confidence .................................... 52
  6.5 Common Sense ........................................................ 53
  6.6 Other Sensors ........................................................ 53

7 Conclusion ............................................................... 55
  7.1 Extensions to Doppelgänger ....................................... 56
    7.1.1 The Interface ...................................................... 56
    7.1.2 Performance ..................................................... 57
    7.1.3 Knowledge and Inference Mechanisms .................. 57
  7.2 Future Work in User Modeling .................................. 58
List of Figures

4.1 The Doppelgänger User Modeling System. ......................... 31
Chapter 1

Introduction

1.1 How To Read This Thesis

This introduction is meant to be complete in itself. Casual readers should read the overview first and then decide if they want to read the rest of the introduction, or better yet, the rest of the thesis.

User Modeling: The Field contains a review of user modeling, placing Doppelgänger in its research context. The Future of User Modeling discusses limitations in the field, identifying areas for improvement. The Doppelgänger System describes the system developed for this thesis, which addresses these areas, and the Implementation chapter explains how it works. The Issues chapter discusses interesting topics concerning the potential impact of user modeling, such as the implications of user modeling systems on privacy. Finally, the Conclusion chapter addresses both short- and long-term extensions to Doppelgänger and implications for future user modeling research.

1.2 Overview

Computers are handicapped by lack of knowledge about their users. Users vary immensely in what they know, what they want to do, their experiences, and their misconceptions. When people deal with other people, they take these characteristics into account. So should computers. Unfortunately, doing this perfectly is as impossible for computers as it is for humans.

A system for maintaining knowledge about individuals, called a user modeling system, has been developed. This system, Doppelgänger, has five primary functions:

- Monitoring sensors that monitor people
• Making inferences from this data
• Incorporating these inferences into distinct models of individuals
• Generalizing about populations of users
• Providing this data to applications

External applications—perhaps a personalized newspaper, or a movie selection assistant—send requests to Doppelgänger. They might ask if a user follows baseball, or if she likes horror films. If Doppelgänger had previously inferred this information, it answers the query. Otherwise, it makes an educated guess based on generalizations from its population of user models.

1.3 The Need: Personalization

As computers become increasingly integrated into home and work environments, the need for knowledge about users becomes paramount. Much of computer science has been dedicated to providing computers with "human" abilities: things that are the same from person to person, such as vision or natural language. Unfortunately, an equally compelling concept has been forsaken: teaching computers about what is different from person to person. Doppelgänger enables this personalization by providing information that can be used to tailor applications for each user.

Many programs bill themselves as “personalized.” That usually means they support customization by user-initiated menu selection; i.e. the user explicitly activates or deactivates preset options. This is user modeling of a sort, but it has two serious drawbacks. First, it is too intrusive. Customizable systems should use human-provided information when available, but when such information is not available, they should exhibit “graceful degradation” and ascertain to the best of their ability what the user wants. The second drawback suffered by this explicit personalization is that all the burden is placed on the user. People make many tacit assumptions when interacting with others; computers should do the same.

Once gathered, information about individuals has numerous uses. Electronic newspapers, desktop managers, advisory systems, and video games are examples of applications that could make good use of this information. All interactive computer applications can benefit in some way from increased knowledge of who is using the application. As shown
later, an electronic newspaper is a concrete example of the desirability of personalization: executives receive a different paper than artists, and even get a different paper depending on whether they're at home or work. Suppose overnight two events occurred: the United States invaded Iraq, and the reader's boss caught the flu. The lead story of the personal newspaper might be "Nine a.m. meeting cancelled," displacing the report on the invasion. And of course, all these newspapers would be substantially different from a nine year old's.

1.4 The Solution: Doppelgänger

Personal information is clearly useful. The question remains: how can this information be obtained? No computational source will provide all the information necessary to make a complete model of a person. After all, this information isn't explicitly available to people either; they make a multitude of inferences from superficial data in a variety of communication channels, such as what a user is wearing, if she is smiling, or the volume of her voice. Any successful user modeling system must do the same.

A user modeling system, called Doppelgänger, has been developed. A Doppelgänger (from German; literally, a double-walker) is a shadowy replica of a living person. This is what user models should be—ethereal copies of a person that progressively become more fleshy, but must always fall short of the real thing.

What should user models contain? An accurate model of the user includes knowledge, beliefs, goals, plans, schedules, behaviors, abilities, preferences, and misconceptions. In short, everything that might be used to predict a user's actions or customize the actions of a computer for that person.

Collections of information about the user are often termed "personality profiles." For Doppelgänger's purposes, this term is misleading; it implies a summary of traits rather than an exhaustive simulation. The term "user model" suggests a more thorough coverage.

Doppelgänger continually updates and refines its models. This is necessary for two reasons. First, the user model should change to reflect changes in the user, whether long term changes (the user becomes an expert in some domain and thus does not require tutelage), or more ephemeral changes (recognizing that the user is going home for the day and consequently needing a reminder to buy milk). Second, the user model should become increasingly more complete as it learns more about the user over time.

The Doppelgänger system consists of four modules: sensors, observers, deducers, and inducers. Sensors are hardware or software that collect data about individuals. Observers
poll the sensors and make *prima facie* inferences from this data. *Deducers* deduce "little facts" from patterns of user activity and incorporate these into the user model. Finally, *inducers* infer generalizations made possible by inclusion of new data. These modules communicate in an *operator language*, which is a restricted set of verbs and prepositions for specifying user actions. Doppelgänger acts as a "personality server," providing client programs with personalization information. These requests can be in simplistic English, which is handled by a primitive natural language parser, or in Doppelgänger's operator language.

Partial support for eight sensors are available in the current implementation of Doppelgänger. Sources of data range from information rich sources, such as any text written by the user, to information-poor sources such as a sensor that monitors idle time at a computer console. Newspace, the Electronic Publishing Group's electronic newspaper, UNIX information, and the Media Lab Badger system, which keeps track of physical user locations, are all sources.

These are only a small subset of the sources potentially available now to Doppelgänger, and a much smaller slice of the sources potentially available to a user modeling system. Advances in vision or natural language enhance sensors useful to Doppelgänger; eye-trackers [CS90] and speech affect recognition [Cah90] are likely to be used in the near future. There is a plethora of information hidden in many of the actions of people working in today's computational environment. An example is electronic mail subscriptions: by looking at what mailing lists people subscribe to, Doppelgänger can make several unobvious inferences. In addition to the subscriptions themselves, the interests of the user can be inferred, as well as the fact that the user reads a lot of electronic mail, and so on.

Today's inferencing mechanisms also fall far short of their potential. What we can infer from available data is a tiny fraction of what could potentially be inferred. Doppelgänger infers information about the user from many sources, some accurate and informative, others error-prone and unenlightening.

Much of the planning that must go into the construction of any user modeling system is how to deal with incomplete information. Doppelgänger compensates for incomplete information by making educated guesses from the universe of user models. If an application asks for the lunchtime of some user, Doppelgänger makes a guess based on the lunchtime of other users sharing attributes such as "workplace" with the first user. How would it know that workplace is a good indicator? It would assume this if a correlation was found between workplaces and lunchtimes. Thus, Doppelgänger makes sophisticated inferences without an
explicit knowledge base; it bootstraps its own knowledge base from the collective personal information of a large number of users.

Given the fallibility of the sensor data and the strong demands put on the inference mechanisms, it is inevitable that mistaken inferences will be made. Doppelgänger minimizes the impact of wrong inferences by building into its inference mechanisms a dynamic confidence rating for all inferences and sensors. No incorrect inference is ever etched in stone because nothing in Doppelgänger is ever etched in stone.

Any application utilizing information provided by Doppelgänger is likely to receive user feedback, which can in turn be relayed to Doppelgänger as an indication of the reliability of its sensors and inference mechanisms. Doppelgänger is designed to make use of this feedback when available.

Due to the sensitivity of the personalization information, users will want access to their model, to learn what the system has discerned about them. A critical part of Doppelgänger is the interface which allows users to peruse their model. Using this interface, users detect and correct wrong information, and can also discover patterns in their behavior of which they were unaware. This modification interface is itself a source of information for Doppelgänger: by observing what parts of the model the user modifies, Doppelgänger can make inferences about the reliability of its sensors, the autoepistemic knowledge of the user, and the efficacy of its inference mechanisms.

An intriguing aspect of user model maintenance is the ability to make use of others' models. Personal newspapers are an example: user models allow readers to view the world through others' eyes. Readers can switch entirely to another's model, or can interpolate between two models. Furthermore, readers can extrapolate beyond two personalities, or even caricature their own personalities. Individual attributes can also be directly controlled. Fabricated models can be constructed to produce extremist newspapers in which one or a few traits are held to be of maximal importance. These are the first steps to automatic manipulation of more complex qualities such as political bias or depth of analysis in news.

Few user modeling systems possess even one of these features, and until now, none possessed all of them. Doppelgänger is an ongoing research effort integrated into a computational environment, continually amassing, inferring, and serving personalization information. This thesis addresses the need for these features, their implementation in Doppelgänger, and their implications.
Chapter 2

User Modeling: The Field

2.1 Overview of the Field

Doppelgänger is a user modeling system. However, it differs from other user modeling systems in a number of ways. To put Doppelgänger in its research context, this chapter discusses the history and current state of user modeling.

Briefly, user modeling can be defined as simulating a computer user. No simulation is perfect; no user modeling system attempts to model all aspects of human behavior. Systems differ in which aspects they emulate.

Most have as a primary or secondary goal plan recognition, i.e. determining what the user hopes to accomplish by using the system. Some restrict themselves to domains in which learning occurs, and attempt to detect and correct user misconceptions; these are called student modeling systems. Other systems focus on inference mechanisms: how stereotypes or defaults are obtained and used. Some address how natural dialog with the user can be enhanced by use of a user model. Finally, a small number concentrate on methods of data acquisition.

The largest use of user models to date has been in intelligent tutoring systems, which use models to represent the student’s understanding of some subject. While user models have been successful in this domain, they have remained entrenched in research: no marketed products include user modeling systems. Research in user modeling has grown exponentially since Rich’s 1979 GRUNDY system, which made book recommendations based on an explicit representation of personality traits [Ric79], and Allen et al.’s 1978 papers on speaker intentions and dialog processing. There is now a user modeling conference [Wah90], a journal of user modeling and user-adapted interaction [umu], and a book devoted to user models in dialog systems [KW89].
2.2 The Scope of User Modeling

2.2.1 What Constitutes a User Model?

User modeling is a nascent field. Most research originated in two subfields of artificial intelligence: dialog systems and formal planning. Dialog systems research concerns how to establish intelligent discourse between computer and humans. Formal planning addresses how computational entities can formalize their goals in such a way that a course of action toward these goals is made computationally tractable.

To say a system contains a user model is to imply that it possesses an explicit representation of one or more users. Implicit models, those which have assumptions about the user integrated into the design, are less interesting: for instance, cars have implicit models because their design contains assumptions about their users.

Kass and Finin contend that modeling user knowledge of the application domain can take on two forms: overlay models and perturbation models [KF88]. An overlay model is based on the assumption that the user's knowledge is a subset of the domain knowledge, and stereotypes about users are hardcoded into the system. An overlay user model can thus be thought of as a template "laid over" the domain knowledge base. Domain concepts can then be identified as known or unknown, reflecting beliefs inferred about the user. Because of its simplicity, overlay modeling is an attractive technique. However, it makes the assumption that user knowledge will always be a subset of the system's knowledge, and that users will organize their knowledge similarly, but this won't always be the case.

The perturbation model is capable of representing user beliefs that the overlay model cannot. Perturbation user models assume that user beliefs are similar to the system's knowledge, but that the user might hold some beliefs that differ from the system's. These differences in the user model are viewed as perturbation of the domain knowledge base. The perturbation user model allows for some deviation in the structure of that knowledge.

Kass and Finin [KF88] identify four important features of explicitly coded models. First, they contain a separate knowledge base. Information about users is collected in a separate module rather than distributed throughout the system. Second, this knowledge must have an explicit representation, one powerful enough to provide a set of inferential services, allowing some implicit knowledge of users that can be automatically inferred when needed. Third, support for abstraction; ways to describe abstract as well as concrete entities. For example, the systems might be able to discuss classes of users and their general properties.
as well as individuals. Fourth, multiple use; since the user model is explicitly represented as a separate module, it can be used in several different ways (e.g. to support a dialog or to classify a new user). This requires that the knowledge be represented in a more general way that does not favor one use at the expense of another. It is highly desirable to express the knowledge in a way that allows it to be reasoned about as well as reasoned with.

2.2.2 User Modeling Dimensions

How should user models be distinguished? Researchers have proposed several scales along which user models are varied.

Finin and Drager made two user modeling distinctions: between models of individual users and models for classes of users, and between long- or short-term models (the temporal extent of the model). Sparck Jones adds a third modifiability dimension: whether the model is static or dynamic. Static models do not change once they are built, while dynamic models change over time.

Rich proposed these same three dimensions, but treats the modifiability category somewhat differently [Ric83]. Instead of static models, she users the term explicit model for one defined by the user that stays the same for the course of a session. Examples of her explicit models are “login” files or customizable environments. She uses the term implicit model to describe models that are acquired during the course of a session and are therefore dynamic.

Three other modeling dimensions are of interest: the method of use (either descriptive or prescriptive), the number of agents (modeling one agent may depend upon the models of other agents as well), and the number of models (more than one model may be necessary to model an individual agent). Most of the natural language systems that focus on inferring the goals and plans of the user use a single, generic model. Finally, Gegg-Harrison proposes an ability-capability distinction—what the user could do as opposed to what the user can do—as an additional dimension to distinguish user models [GH90].

2.3 Research Areas in User Modeling

The Second International Workshop on User Modeling divided the field into six categories: psychological foundations, student modeling, plan recognition, theoretical issues, user modeling in natural language dialog, and user modeling for natural language generation.

A conceptually cleaner division is:
1. Student Modeling

2. Plan Recognition

3. Inference Mechanisms

4. Dialog Systems

5. Data Acquisition

This section duplicates this organization and explores each area in depth. The placement of each work inside one section is somewhat arbitrary since systems often overlap areas.

2.3.1 Student Modeling

Several researchers have focused their attention upon the task of modeling students. Students possess predictable misconceptions, which they presumably want corrected, and these misconceptions can be identified and corrected through natural language dialog. Systems using these techniques are often called intelligent tutoring systems (ITS).

Gegg-Harrison proposes a framework for classifying intelligent tutoring systems based on why the student model is needed, what is being modeled, and how it is being modeled [GH90]. This distinction is made in his system, which models the student’s cognitive potential in a schema-based Prolog tutor.

McCalla and Greer also consider the problem of student modeling in an ITS [MG90]. They focus on tracking a student’s changing knowledge. First, they overview the main problems faced by an ITS: representing incorrect as well as correct perceptions of the world, handling the wide variety of knowledge the student may have, understanding the complexities of student behavior, and maintaining the student model as the student’s knowledge (and the tutoring system’s perception of this knowledge) evolves. Then, they focus on the student model maintenance problem, and describe some attempts to extend standard truth maintenance approaches to handle student model maintenance.

Quilici [Qui90] developed a user modeling system that models student learning in the UNIX operating system. Rather than focusing on recognizing user plans, he focuses on
inferring user beliefs about those plans and goals. He observes that an automated advisor must be able to construct a model of the user's relevant plan-oriented beliefs and reasons for holding them, as well as determine which beliefs are mistaken and select advisor beliefs to present as a corrective response. His approach relies solely on domain-independent knowledge about planning, rather than on domain or task-specific rules, allowing the advisor to recognize and respond to novel user misconceptions.

London notes that intelligent computer-aided instruction (ICAI) systems have sought increased flexibility to respond appropriately to the multifaceted interests of students [Lon90]. His GUIDON2 and IMAGE systems provide a wide range of instructional approaches, enabling the flexible communication needed. The IMAGE student modeling component emulates a skilled teacher's responses, in terms of type, content, and viewpoint of advice. The GUIDON2 system supports the multiple instructional approaches necessary.

2.3.2 Plan Recognition

User modeling researchers contend that critical to the user modeling task is plan recognition, that is, identifying what users are trying to accomplish and how they are trying to accomplish it.

Cohen et al. use a perturbation model, allowing user plans to differ from the system's library of fixed plans [CSSvB90]. They provide procedures for handling two distinct extensions: interpreting tense information from the user and admitting novel information.

Wu discusses the implications of active user model acquisition upon plan recognition, domain planning and dialog planning in dialog architectures [Wu90]. He contends that for these to be attained, the user needs to be explicitly queried during the course of the dialog. Wu contends that this capability requires major extensions to existing plan recognition, domain planning, and dialog planning models, and proposes a framework for handling this.

Carberry presents a process model of plan recognition that includes strategies for making rational default inferences and for updating the model of the user's plan as dialog progresses [Car90]. The process model can be explained and justified to the user when discrepancies arise between it and what the user is actually trying to accomplish. When contradictions are detected, the contradicted goals are removed, and an attempt is made to identify the reasons for the contradiction. Like Wu, she concedes that in some cases, a "subdialog" will be necessary.
Eller and Carberry present a meta-rule approach to deduce the cause of errors from these discrepancies [EC90]. Meta-rules relax the plan inference process, enabling consideration of alternative beliefs about the user's plans and goals. When the system has difficulty assimilating an utterance into its model, the meta-rules relax the user's utterance, the system's processing rules, and the system's model of the user's plan. Multiple hypotheses are generated by relaxing the interpretation process to guide a negotiation dialog with the user, during which the participants identify the cause of the suspected dialog ill-formedness and isolate and repair errors in the model. This approach provides a unified framework for handling both well-formed and ill-formed dialog, avoids unnatural interpretations when the dialog is proceeding smoothly, and facilitates a nonmonotonic plan recognition system.

Quilici et al. discusses the problem of recognizing and responding to misconceptions in advice-seeking dialogs, concentrating on the problems of novice computer users [QDF88]. In their system, responses both correct the user's mistaken belief and address the problematic user beliefs that led to it. Responding appropriately to a potentially incorrect user belief is presented as a process of: (1) checking whether the advisor holds the user's belief; (2) confirming the belief as a misconception by finding an explanation for why the advisor does not hold this belief; (3) detecting the mistaken beliefs underlying the misconception by trying to explain why the user holds the incorrect belief, and (4) providing these explanations to the user. An explanation is shown to correspond to a set of advisor beliefs, and searching for an explanation to proving whether different configurations of advisor beliefs hold.

2.3.3 Inference Mechanisms

Because of its origins and its youth, user modeling retains strong ties to artificial intelligence (AI). Work on inference mechanisms for user models includes applying AI inference techniques to user modeling, developing inference mechanisms specific to user modeling, and establishing theoretical frameworks for user modeling.

Many issues that arise for user modeling are addressed by formal AI research, which has provided some useful tools. For example, user modeling attempts to build and maintain a detailed model of a user based on a small set of observations about the user. Abstractly, the problem is how to determine what is true about the user despite incomplete information about that object. AI has addressed this problem by making assumptions about the object, a process called default reasoning [vA90].
Van Arragon speculates that user modeling research can benefit from formal automated reasoning tools, but that these formal tools often need to be modified to suit the needs of user modeling. Theorist, an AI framework for default reasoning, can be used as a tool for building and maintaining a model, and as a model of a user's default reasoning. To apply Theorist to both tasks, he developed Nested Theorist (NT), a simple tool based on Theorist that allows default reasoning on arbitrarily many levels.

Appelt and Pollack use abductive reasoning to create user models [AP90]. They use a type of abductive reasoning called weighted abduction for comparing assumptions, and by this method inferring aspects of an agent’s mental state given observations involving beliefs and intended actions.

Ballim and Wilks show how distributed stereotypical beliefs, coupled with a tangled hierarchy of agents and stereotypical classes of agents, can be used to dynamically ascribe beliefs [BW90]. They consider variations on the notion of belief ascription that allow creation of beliefs about composite stereotypical classes of agents, e.g. a medically informed physicist. This flexible approach permits easy modification of the model of any given user as further information becomes available.

Kay’s um system provides a user modeling architecture that contains a collection of rule-based tools for acquiring and maintaining reusable user models [Kay90]. The underlying philosophy of um is to view the user as the owner of her user model and as a cooperative agent in the construction of their user model. In keeping with this philosophy, the environment includes tools to cooperatively elicit modeling information. One such tool asks the user to externalize knowledge in an arbitrary domain—this gives models of their deeper conceptual understanding as well as more superficial and syntactic aspects of their knowledge.

Bunt describes a formalism for modular, incremental modeling of the beliefs and intentions of the human user of a intelligent interactive information system [Bun90].

### 2.3.4 Dialog Systems

A significant proportion of user modeling researchers are investigating the use of user modeling for dialog systems. Their research concerns both natural language input and output: what cues are available from natural language input, and how computer-generated natural language can be tailored to the user.
Dialog Input

Brennan notes that in human conversations, people design what they say specifically for their conversational partners [Bre90]. Keyboard conversations, whether between humans or between human and computer, follow many of the same conventions, but differ in some ways, such as the amount of background knowledge assumed. Style shapes what people type, and among other things, people expect their partners to maintain connectedness across conversational issues.

Kass’ GUMAC (General User Model Acquisition Component) acquires user models from dialog using implicit acquisition techniques such as computing presuppositions or entailments [Kas90]. GUMAC makes rule-based default inferences about user beliefs from their interaction with an advisory expert system. Kass bases his rules upon features of human action and conversation that constrain people’s behavior and establish expectations about their knowledge.

When people spot others’ misconceptions and respond to them, their responses generally provide reasoning refuting the misconception. McCoy replicates this behavior using a highlighted model, one in which attributes critical to a misconception are marked as such [McC88]. She investigates how domain perspective can be modeled to provide the needed highlighting and introduces a similarity metric that is sensitive to the highlighting provided by the domain perspective.

McKevitt showed relationships between natural language queries in the UNIX domain [McK90]. In his study, students interact with a computer through typed dialog at a monitor and are told they are conversing with a computer. He concludes that speech acts are not only useful for analyzing the intention of subjects in dialog and hence their goals and plans, but also for showing user modeling data and that they have implications for user models. His analysis permits recognition of users bringing terminology from another operating system into UNIX.

In an issue of Computational Linguistics devoted to user modeling, several researchers focus on what, if any, difference exists between discourse models and user models. There are three positions taken: first, that the discourse model is a part of the user model ([Sch88] and [Kob88]), second, that the discourse model intersects the user model ([Chi88]), and third, that the discourse model and the user model are distinct ([Wah88], [Coh88], [Mor88], [Jon88]). To summarize, and do injustice to, all of these papers, a discourse model is a structure detailing information about objects referred to in a conversation as well as the linguistic structure, the attentional state, and the intentional structure.
User Modeling for Natural Language Generation

Paris describes TAILOR, a question answering program providing access to a large amount of data that tailors its answers to each user [Par88]. The user's level of knowledge about the discourse domain is an important factor in this tailoring. Descriptions are more parts-oriented for less knowledgeable users, and more process-oriented for users with more expertise. In TAILOR, these strategies are mixed for users lying between these two extremes to include the appropriate information from the knowledge base.

Another question answering system, called GENIE, was developed by Wolz [Wol90]. GENIE generates text that responds to user questions and attempts to teach the user when convenient. It has a three-part user model that includes a situational context, a discourse context and a model of user domain knowledge. The model affects the behavior of GENIE "experts": the observational expert, the domain expert, the analytic expert and the explanatory expert. While Wolz's system exploits a user model, she provides no advice on how to actually build it.

In the process of generating text, competent speakers take into consideration the effect their utterances are likely to have on listeners [Zuk90]. In particular, they try to anticipate and prevent possible comprehension problems. Zukerman's WISHFUL system incorporates a mechanism which emulates this behavior for the generation of rhetorical devices that are instrumental to the attainment of a communicative goal. WISHFUL anticipates the effect of a message on a model of a listener's beliefs, and proposes rhetorical devices to preclude possible adverse effects.

Sarner and Carberry present a mathematically-oriented strategy for reasoning on a multifaceted user model to generate definition tailored to the user's needs [SC90]. Their strategy considers the user's current focus of attention in the partially constructed plan, the user's domain knowledge, and her receptivity to different kinds of information. This strategy generates definitions that appear natural and represent cooperative, intelligent behavior.

Maybury's TEXPLAN (for Textual Explanation Planning) produces personalized text output by reasoning about the rhetorical structure and pragmatic function of text, guided by both a user model and system knowledge [May90]. Maybury formalizes communicative plans that produce user-oriented text by reasoning about a system's model of the user's knowledge, beliefs, abilities, goals, and plans.
Haimowitz’s SERUM system tailors dialog in the domain of lung disease for AIDS patients [Hai89], [Hai90]. SERUM combines two hypotheses. First, that the dialog system should explicitly model both the person directly interacting with the dialog system (the agent) and the person reasoned about by the expert system (the patient) in order to communicate meaningfully with both people. Second, that a dialog system can model the domain-related beliefs, preferences, and concerns of both its users and generate responses empathetic to both. SERUM does this by first converting domain knowledge and agent and patient properties into pragmatic objectives, e.g. empathy. Then these objectives are transformed into surface structure cues such as object emphasis and level of technicality. Finally, these cues are used to generate text that is natural, appropriately technical, and downplays information that is unpleasant or undesirable to the agent or patient.

2.3.5 Data Acquisition

User models are traditionally deduced from keyboard discourse, or, rarely, “pointing” with a mouse [CS90]. In contrast, Crosby and Stelovsky use eye movement analysis for user models. They had three goals: to analyze and classify subjects’ viewing patterns, to explore the effects of errors on viewing patterns, and to examine whether the time spent viewing critical program areas predicts user expertise. In their experiment, computer science students were asked to read a computer program. Resulting eye movement patterns were used to distinguish the students by experience, comprehension, and cognitive style.

2.4 User Modeling in News

2.4.1 Electronic Newspapers

Pervading the research of the MIT Media Lab Electronic Publishing Group is the recognition that there will soon be high-bandwidth communication channels entering everyone’s home. When this occurs, it will permit manipulation of both the signal and content of traditional mass media [LB87], [BC88].

Several Electronic Publishing Group projects concern personalization. Salomon developed an Electronic Special Interest Magazine [Sal86]. It tailored the content of the magazine to the user based on a profile of user actions. Newspeek and its successor, Newspage, are personalized electronic newspapers. Newspage makes use of Doppelgänger to personalize both content (which articles are shown) and presentation (how they are shown). More information on Newspage is in the Implementation chapter.
Allen was the first to formally explore using user models for news [All87]. He sees user models as providing a "personal assistant" to aid article selection. People may have a bias against being controlled, or they may feel that interaction with a computer is dehumanizing. A related problem is that people may feel self-conscious about the articles based on their interests that are selected, e.g. if the user model frequently suggested sensational news articles and the reader had objections to that type of material. Conceivably, this could cause a rejection of the news articles and of the system itself. Jones [Jon86] has distinguished between the efficiency, the effectiveness, and the acceptability of user models. On the other hand, in some cases interaction with a computer has been found to be preferred to interaction with a human being.

Another concern is that user models will often be too narrow because of their mechanistic approach. In addition, readers might miss the advantage of serendipitously useful information. For instance, many people report that a seemingly irrelevant news item has become of great importance after they read it. However, concrete evidence for this effect is weak, and it might as easily be argued that exposure to articles selected by some criterion might produce more 'serendipity' than randomly selected articles.

2.4.2 Information Retrieval

As Allen points out, information retrieval involves obtaining documents which are relevant to a specific user query. Thus, the emphasis is on characterizing documents rather than people, and inferences are made primarily along pre-determined dimensions of document similarity. Nonetheless, in some cases there is a fine line between information retrieval and user modeling. For instance, feedback procedures change the direction of a query based on user responses. Selective Dissemination of Information (SDI) systems match a stored user query against new documents as they are entered into a system. SDI systems are usually keyword driven and much of the work has emphasized techniques for enhancement to these keyword interfaces. Finally, some proposals have been made to incorporate user models with information retrieval interfaces [BDB85].

An information retrieval system that has recently become popular is Thinking Machine's Wide Areas Information Server (WAIS). Developed as a joint effort among Thinking Machines, Dow Jones, and Apple Computer, the WAIS provides a robust framework for the maintenance of a central database of news [Kah89]. DowQuest, which uses a WAIS, maintains news "agents" that act on the reader's behalf, watching for news that might be of
interest. Newspace could well make use of such a server for its storage. Unfortunately, its protocol restricts it to text, and so does not address multimedia issues, but provides a glimpse of what will happen when Newspace begins to store and access the immense quantities of information that any newspaper requires.
Chapter 3

The Future of User Modeling

This chapter is composed of two sections. The first discusses the trends in user modeling and other fields, and the second marks the transition for the rest of this paper into personal work, opinions, and speculation.

3.1 Trends

3.1.1 Current Trends in User Modeling

The short history of user modeling has set some clear directions for future research. Some of these directions are unarguably advances; the obstacles have been technical, not philosophical. An example is the maintenance of individual user models. This section discusses some of the more arguable trends.

Generalized User Modeling Systems

Alfred Kobsa of the University of Saarlandes suggests that the next step in user modeling is user modeling shell systems [Kob90], [Kob91]. Just as expert system shells generate expert systems when provided with application-specific data, user modeling shell systems are sets of tools that generate a user modeling component for an unspecified application domain.

Kobsa's BGP-MS (for Belief, Goal, and Plan Maintenance System), is such a system. BGP-MS becomes a user modeling component when provided with application dependent stereotypes. It provides a powerful representation language for conceptual knowledge, a graphical interface for the user model developer, and a functional interface for accessing and updating the model of the current user, as well as domain independent inference mechanisms, support for user-defined inferences, and a stereotype manager.
Finin's GUMS (General User Modeling Shell) provides facilities for a stereotype hierarchy, default reasoning, and domain-specific rules. Questions such as whether the application acquires knowledge about the user by explicit interview or by "eavesdropping" are left up to the application. Systems such as GUMS purportedly provide the bridge between user modeling as described in the other chapters and wide applicability in the "real world".

Data Acquisition

Practically all user modeling systems, including those discussed just above, are closely tied to planning and natural dialog. The need for user modeling arose in part out of the complaint that when people engage in discourse with computers, the computers "didn’t infer the right things about them."

Nearly every user modeling system is related to natural dialog: they all focus on acquisition of a user model through natural language or tailoring natural language output with a user model. The only exception is the eye-tracker used for data acquisition by Crosby and Stelovsky [CS90].

Plan Recognition and Generation

An increasing amount of research is being directed toward recognizing the plan of the user, and toward generating plans believed to be held by the user.

3.1.2 Planning

Given the amount of user modeling research addressing planning, it is helpful to look at more unorthodox approaches to planning. In particular, the notion of situated action merits consideration. The central tenet of situated action is that humans don't plan; they improvise. Rather than consciously or unconsciously plotting all their actions explicitly, they rely on learned cues to guide them through both familiar and unfamiliar situations.

Situated action is realized through a deictic representation, which means that information is maintained relative to an agent's current state. This idea is not a new one, dating back to early planning literature in the 1960's. Such a representation started to garner support when Lucy Suchman wrote Plans and Situated Actions, which provided some powerful arguments for this representation [Suc87]. Suchman and Agre observed that traditional planning theory has drawbacks; it considers the world to be a place of continuous hazards.
Agre advances this notion, observing that conventional planning treats human activity as the construction and execution of computer-program-like structures, leading to “grossly impractical technical proposals” [Agr88], [AC88], [Agr85]. Agre views contingency as a central feature of everyday activity and improvisation is the principal means by which people get along in the world. Agre’s model structures improvised activity as running arguments that continually redefine what to do, and he uses a deictic representation to treat things in the world in their relation to the agent and not as self-contained objects.

Maes views agents as a collection of competence modules [Mae89]. Action emerges from interaction between these modules. Maes’ agents can smoothly interpolate between formalized goals and situated action, bias toward ongoing plans (inertia) for adaptivity, thoughtfulness for speed, and adjust its sensitivity to goal conflicts. Maes’ work is inspired by Minsky’s Society of Mind [Min86], which shows how intelligent action arises from the interplay of simpler agents.

3.2 Problems with User Modeling

This thesis has presented where user modeling has been, where it is now, and where it is going. Now it’s time to complain. This section addresses how current user modeling systems can be improved upon.

3.2.1 Data Acquisition

When surveying the user modeling literature, one is struck by the extent to which input is restricted to dialog. With the exception of [CS90], all user modeling systems rely upon an extremely narrow communication channel: the keyboard, and these researchers are far from fully exploiting even this limited medium. The mistakes people make when typing, the speed of typing, how forcefully they strike the keys, and the pauses between the keystrokes all individuate people. At first glance, this information seems worthless. Those who think so should note that this information was used in the MIT Media Lab to verify user identities without passwords. Building a user model solely from which characters are typed at the keyboard is like marrying a pen pal you’ve never seen.

Data is conveyed through innumerable channels. There are conventional channels, those that humans receive through their senses, and computational channels, information that is made available as a result of tasks performed in a computational environment, such as desktop managers that contain lists of important things to do, or electronic newspapers
that can indicate what a user's interests are. Capturing this data raises additional issues, such as the logistics of data acquisition and the potential for privacy violations, but offers immense benefits that can't be realized with keyboard discourse. And yet, a special issue of Computational Linguistics contained essays from eight leading user modeling researchers discussing the relation of user models to discourse models, and this was a major point of contention at the First International Conference on User Modeling.

3.2.2 Planning Revisited

In the last section, trends in planning for user modeling were mentioned, as well as trends in planning apart from user modeling. While the ideas of Minsky, Agre, and Maes have yet to become widespread outside of MIT, they are illustrative here. Their criticisms of conventional planning techniques apply to research in user modeling. Any user modeling system which relies upon a rigid representation of user plans is bound to fail. Only rarely will computers be able to realistically formalize the user's goals; after all, usually the user doesn't even know what her goals are. Furthermore, plans change. Not only on the order of days or months, but every second.

3.2.3 Stereotypes

Kass divides user models into overlay and perturbation models depending on how stereotypes are stored [KF88]. However, there is a third alternative: neither. A user modeling system is neither an overlay nor a perturbation model if it has no explicit storage of stereotypes. Deduced facts can be represented explicitly, while generalizations from these facts are not. In a robust user modeling system, which changes continually, this is necessary. With the wealth of data streaming in, stereotypes change from minute to minute.

3.2.4 Intrusiveness

User modeling should be unintrusive. Many, though by no means all, of the user modeling systems in existence today rely upon computer-initiated human intervention. Little wonder, given that they are trying to infer a user model from just keyboard discourse. Unfortunately, these "subdialogs" can only annoy the user by keeping her from getting her work done.

These subdialogs and other limitations of user modeling systems create a problem: they thrust the user modeling system upon the user's consciousness. The visibility of user modeling is a complex issue; while intrusiveness is by definition undesirable, in some cases it
is important for the user to know exactly how the user modeling system is affecting system behavior.

3.2.5 Role of User Modeling in a Larger System

Kobsa includes a graphical interface for the user model developer [Kob90]; in BGP-MS, developers can use this interface to provide application-dependent stereotypes and create a user modeling component of a larger system.

The idea of a user modeling system as a "personalization server" is a more compelling idea. In such a system, there would be no need for a graphical interface because there would be no user model developer.

A server is more adaptable to different computational environments. None of the user modeling systems described are autonomous in the sense that they can drop into a computational environment and start to process data. They are constrained not by their computational environment but rather by the domain and type of discourse.

3.2.6 Architecture

No user modeling systems facilitate extensions to themselves. The more advanced ones perhaps allow addition of new stereotypes, but allow no other modifications to system operations. This derives in part from the idea that user modeling systems should be abstracted-out components of other, larger systems, that do their job as well as their planning mechanisms let them.

But there is another sort of extensibility that is lacking in this approach, and that is extensibility of the data acquisition methods. To adapt to a computational environment to which new features are always being added, user modeling systems need an extensible architecture, one that not only allows but facilitates the addition of new sensors to the system.
Chapter 4

The Doppelgänger System

This chapter discusses the features and use of Doppelgänger. More technical details are in the implementation chapter.

4.1 Overview

Doppelgänger is a user modeling system that fully exploits the user’s computational environment, supporting arbitrarily many sensors of different media. It is unintrusive, does not mold the user into a rigid plan or belief structure, and has an extensible architecture. Doppelgänger allows people to view and modify their individual models, and allows users to interpolate between or extrapolate from models.

Doppelgänger attempts to live up to its name. There is no magical data source that provides Doppelgänger with everything it wants to know; it does the best it can with what’s available. Likewise, no inference engine provides Doppelgänger with all useful inferences from available data. What the sources and inference mechanisms lack in quality they make up for through quantity, persistence, and feedback.

Doppelgänger runs continuously. It resides on a workstation that is part of a local network made up of UNIX workstations. All means of sensing information about the users are accessible from this network, and client applications send requests for information, and receive replies, over this network.

4.2 Data Acquisition: Sensors and Observers

This section discusses how Doppelgänger acquires the data used to construct user models and what the models themselves contain.
Figure 4.1: The Doppelgänger User Modeling System.
Sensors relay their information about users to observers, which in turn relay their information to the deducer and inducer modules. At bottom, the data maintained by Doppelgänger is "served" out to client applications, which can also serve as sensors themselves. Everything inside the large box is sensor- and application-independent.
As people spend more time in a computational environment, a wider range of activities will be available to sensors such as these. These will include bank transactions, television viewing, phone answering, shopping, and much more.

Doppelgänger has eight methods of sensing user actions. The hardware or software devices that record this data are called sensors, which produce sensations. Each sensor has a corresponding observer which polls its sensor periodically and produce perceptions. Observers make prima facie inferences from the sensor data before passing the data on, via an operator language, to modules later in the system. Doppelgänger resides in a network of machines running the UNIX operating system at the MIT Media Laboratory. While its sensors are specific to this computational environment, the core of Doppelgänger is not. If Doppelgänger were transplanted to a different building, different means of sensing data about the user would be available; data acquisition would be drastically affected, but nothing else.

When Doppelgänger is started, it must be told who to track. This could be avoided; Doppelgänger could just track all users identified by any of its sensors, but privacy concerns prohibit this. Messages can be sent to Doppelgänger telling it to activate particular sensors or tracking of particular people. More selective messages, such as “activate this sensor for this person” or “deactivate this sensor for all but this person” are supported. Doppelgänger’s observers include timestamps with all data, so that sequences and patterns of actions can be identified.

The following sensors are currently available to Doppelgänger:

- UNIX finger

The UNIX finger utility provides information on when users last used a computer system, where they connected from, and what their idle time is. Some less ephemeral biographical information is present, such as full name, office number, and office and home phone numbers.

The finger observer makes some simple inferences from this data, obvious things that anyone familiar with the utility would immediately know. From the standpoint of human intelligence, they are probably too simple to be considered inferences; it is more of a parsing exercise.
- UNIX aliases

The aliases file contains the list of electronic mailing lists to which users subscribe. This sensor parses this file, identifying all the mailing lists for each user.

Parsing this list enables provides more sophisticated inferences than the finger sensors. Information is made available about what sort of mail the user likes to receive, what the user is interested in, and how important electronic mail is as a source of information for the user.

- UNIX lastcomm

The lastcomm sensor uses the UNIX lastcomm utility. Lastcomm, given a username, lists the commands that user has executed and how long each command took to execute.

The observer for this sensor possesses some knowledge of different UNIX commands, and can infer from the sensor output what the user is spending time doing. For instance, if the user has been executing the word processor “emacs” for an hour, it is likely she is composing a lengthy document and doesn’t want to be bothered, whereas the game “rogue” has a substantially lower priority.

- UNIX calendar

The UNIX calendar utility notifies users of upcoming events. It consults the file “calendar” in the user’s home directory, which the user explicitly fills with dates and names of events such as meetings and birthdays.

When the user fills the calendar file, she is indicating which events are of importance to her. Doppelgänger uses this information about the user’s schedule to make guesses about what the user is doing at a given time.

- Badger (MIT Media Laboratory Speech Research Group)

The Badger system tracks people wearing specially designed badges throughout the Media Lab. Location information can then be used for personalization.

Each user wears a badge that emits an infrared bit pattern sensed by wall-mounted receivers that relay their information to a server. A sample use of the Badger information could be the routing of telephone calls: wherever the user’s location in the Media Lab, the closest telephone rings when her office telephone is dialed.
The observer can deduce from the Badger system who a user is near. If the user has something she should (or shouldn’t) do in the proximity of another person that Doppelgänger knows about, behavior can be tailored to reflect this.

- Free Text

This sensor handles a wide range of input, from stream-of-consciousness text written by the user to electronic mail. Doppelgänger can extract some information from unrestricted text—that is, text with no descriptive information to indicate what it’s about. Some text, such as electronic mail, arrive with meta-data that provides aids for interpreting the raw text; in this case the observer has an easier job.

As an example of how Doppelgänger would parse free text, if electronic mail to or from the user repeatedly contains the word “vision”, Doppelgänger will infer that the user is interested in other documents containing this word.

This sensor is weak due to Doppelgänger’s primitive parser. As more sophisticated parsers are developed, Doppelgänger will make use of them, but completely understanding text is not only beyond the scope of this thesis, but beyond the scope of current linguistics.

- Newspace (MIT Media Laboratory Electronic Publishing Group)

The largest client of Doppelgänger is Newspace, the MIT Media Lab Electronic Publishing Group’s electronic newspaper. Newspace uses Doppelgänger data to tailor the content and presentation of each personalized newspaper to its reader. Feedback from the newspaper—which articles and sections the user has read—are relayed back to Doppelgänger.

If the user consistently skips the front page in favor of the sports section, Doppelgänger would infer that the user is interested in sports, and tell Newspace that the front page should include more sports articles. Perhaps later Doppelgänger would notice that not only does the reader read only sports articles, but also prefers basketball to football or baseball. This would then be reflected in the presentation.

- Model Manipulator

This is the most interesting, and perhaps the most important, sensor.
Because of the personal nature of the data involved, and the fallibility of the plan inferencing mechanisms, it is imperative that users be able to view and modify their models. The modification process is itself a source of data to Doppelgänger, providing information about how it is doing.

Users can set the confidence of their assertions. For instance, if a user isn’t sure he likes spicy foods all the time, he can supply a moderate value for the confidence. And if the user is sure he is male, he will set the confidence of that assertion to 100. Because of Doppelgänger’s gradient descent learning algorithm, nothing with a confidence of 100 can be changed; such a fact would be de facto hardcoded.

4.3 The Operator Language

User data is passed from module to module in Doppelgänger’s operator language. This is a restricted set of verbs and prepositions. The design of the operator language was based loosely on Schank’s script work. Schank expressed events using a restricted set of verbs; contending that virtually all actions could be expressed with a small set of verbs [SR81]. An expanded set of these verbs make up the operator language verbs. Each perception has exactly one verb and optional prepositional phrases.

The operator language is designed to be general enough to handle different sensors in the future without modification. Since not all sensors can be anticipated, this broadness is not guaranteed. New verbs and prepositions can be added with a minimum of difficulty, and it’s unlikely they will have to be added anyway, since a good observer will provide the intelligence needed to convert sensations into meaningful perceptions, even given a limited operator language.

4.4 Doppelgänger’s Inference Engine

4.4.1 The Deducer and Inducer

The sensors and corresponding observers described in the previous section are interpreted by the other two modules of Doppelgänger: the deducer and inducer.
The Deducer

When the observers have generated their *prima facie* inferences from the sensors, the resulting perceptions are passed to the deducer. The deducer compares this newly acquired data to what is already in the model. It notices what has changed in the model and marks these as significant. For instance, if a user had been subscribing to the "cyclists" mailing list, and a poll of the aliases sensor did not identify him as a member any more, that change would be entered into the model, along with the conclusion that the cyclists mailing list is no longer of interest to the user. Another inference made would be that the user is less interested in cycling than she was before. Since this is a more tenuous conclusion, the associated confidence level would be smaller. While the observer infers what it can given knowledge of only the sensor, the deducer makes further inferences given the output of the observer and the user model.

The deducer has two different functions, both of which are used on each perception it receives. *Keyword indexing* maintains and searches a cumulative histogram of interesting nouns encountered by the observers. This is most useful for free text, in which no information about how to parse the raw data is provided; although meaning can't be ascribed to the text, some cues as to what it's about can be deduced. *Change maintenance* identifies which data has and hasn't changed. If the aliases sensor no longer reports a user as being a member of a list, the entry is removed from the user model and adds an entry indicating that the user has unsubscribed from the list.

The Inducer

When a client application makes a request for information regarding a user, Doppelgänger searches the user's model. If the information is there, it is given to the application. If it is not, the inducer is activated.

The inducer has the problem of trying to make educated guesses without much of an education. Since there is no library of preset stereotypes in Doppelgänger, it is job of the inducer to generate them. The inducer asks itself, "What if the application had asked the same question about another person?" It asks this for every person it knows about, and has each model cast votes for its answer to the query; the number of votes each model is allowed to cast is proportional to the similarity between it and the queried model.

The higher the number of votes cast, the higher the probability the answer is accurate. Uncertainty is reflected in a confidence rating for each answer generated.
4.4.2 Chains of Reasoning

How does the deducer decide which techniques work best? This is done through feedback from the client applications. Every deduction and induction made carries with it a record of the chain of reasoning used to arrive at the inference. The rules themselves are stored, as well as the reasons for the rules.

Given the amount of work Doppelgänger performs for each induction, this rapidly becomes an unwieldy amount of information. When a user asks to view the chain of reasoning behind an assertion, she is presented with all the rules that led up to the assertion, which is dangerously close to the entire history of Doppelgänger. This information is arbitrarily truncated so as not to overwhelm the user, but doing so can hide critical rules from view. This is a problem: only the important rules should be kept, to avoid Doppelgänger cluttering its resources.

4.4.3 So Where’s The Knowledge?

Doppelgänger is neither an overlay model nor a perturbation model, as Kass and Finin declared all user modeling systems must be [KF88]. This is because no knowledge is hardcoded into the system; it is turned on tabula rasa and learns from that point on.

Most systems hardcode stereotypes, but this precludes modification of them— they should be dynamic. No stereotype is perfect; otherwise, it wouldn’t be a stereotype, it would be a fact. Doppelgänger contains no explicit stereotypes, only implicit educated guesses. They are automatically generated when requested (directly or indirectly) by an application. This is done because new information which could potentially affect these stereotypes is being gathered continuously; rather than update a library of stereotypes every time a new fact is obtained, the stereotypes are inferred when needed.

4.5 Clients Using Doppelgänger Data

4.5.1 Accessing the Data

Requests for information are sent to Doppelgänger by external applications acting as clients. These are processed through the command line. Requests can either be made in the operator language, or in simple English.

Operator language:
doppelganger -q 'user Jon attribute topology query interest'
⇒ 90 100 85 100

Natural language:

doppelganger -q 'Does Jon like topology?'
⇒ 90 100 85 100

doppelganger -q 'Is topology liked by Jon?' (this is unparseable input)
⇒ 0 0 0 0

Each command has the same effect; a string is sent to standard output. The string
has four numbers: the strength of the assertion (how much Jon likes topology) with the
maximum possible strength, and the confidence that the strength value is correct with the
maximum possible confidence.

4.5.2 Client Feedback, or "How Am I Doing?"

Learning is made possible as a result of the feedback provided by the client applications.
The way each inference "voted" is remembered, and is used to punish or reward each one
after the user’s response has been determined. A gradient descent algorithm is used to
adjust both strength and confidence values for each entry that contributed to the decision
to include or exclude the article from the user’s perusal. A high confidence value dampens
the strength value, preventing it from changing drastically, and a low confidence value
permits the strength value to sway widely. For the feedback provided by Newspace, the
strength has converged to a steady state within a few hundred iterations— with each article
as an iteration.

Since there are theoretically no constraints on the input Doppelgänger receives, it can’t
be shown that these values will always converge to a steady state. If a user tried to “fool”
Doppelgänger, for instance, the values might never converge. Even if they don’t, and
oscillate about one or more values, no harm is done: Doppelgänger’s choice won’t stray far.

4.5.3 Extensible Architecture

Doppelgänger is modular. Each sensor requires its own observer, but after this stage in
the processing, all data is treated the same. The deducer and inducer act independently of
anything else, obviating the need for any hardcoding (save the sensor and observer) when
new methods of data acquisition are created.
Thus, new sensors can be added with ease. A sensor template and an observer template are provided for developers who might be adding their own sensors. This is what is meant by extensible architecture—Doppelgänger can adapt to different computational environments by swapping sensors, and can grow to handle enhancements to the environment because of the ease with which sensors can be added and the common language they all speak.
Chapter 5

Implementation

This chapter discusses the technical details of Doppelgänger. It assumes an understanding of the features and use of Doppelgänger discussed in the previous chapter.

5.1 Context

This thesis grew out of the Newspace project, part of the MIT Media Lab Electronic Publishing Group [BLO+91]. Newspace is a personalized, multimedia, display-independent newspaper. It continuously collects online news from a multitude of sources, ranging from impersonal mass media sources such as the Associated Press wire to highly personal sources such as electronic mail. Newspace is a client of Doppelgänger, submitting requests for information about a person to it for the purpose of filtering news articles. The reader of the newspaper is presented with a few dozen news articles culled from the thousands that arrive every day.

In everyday life, this plethora of information necessitates filters that expose us to a subset of this “world of news”. Some of the filters are primitive: consider someone whose only source of news is the New York Times. The restriction to just this source is itself a filter. Not only is he excluding literally thousands of other news sources, he’s also not even making full use of the one he has, since he doesn’t read every last word of the Times. Rather, he has some algorithm for searching the newspaper. It’s bound to be imperfect; no one gets exactly the information they want out of the newspaper each time they read it, since they don’t know what they want a priori. Since different people have different algorithms for reading the paper, it’s possible for an individual’s acquaintances to act as filters, calling the user’s attention to certain articles. Incorporating personalization into an electronic newspaper permits better filtering; the entire newspaper will still be there, but
presented in a format that shows the reader more of the news she wants to see, and less of the news that she doesn’t. Newspace accomplishes its filtering through the interaction of three modules, described here:

5.1.1 Presentation

Lye developed the presentation environment for Newspace [Lye91]. Advances in CRTs now permit color displays with resolution of 2000 lines and a screen area of newspaper broadsheet size. Now that the CRT has started to rival paper in legibility, it can be used to present personalized newspapers with a range of new possibilities in content selection, imagery, typography, and human interaction. What the reader sees looks like a passive, conventional newspaper, but it allows interaction; the newspaper watches you read it and acts accordingly.

Newspace has experimented with indicating the importance of each article to the reader in the size of the window, the size of the font, use of color and motion, and other visual attributes.

Since broadsheet displays are not portable, Newspace is also investigating use of portable computers as personalized newspaper platforms. This portability is gained at the cost of screen size, video capability, and similar features. Doppelgänger takes these into account when tailoring content and presentation for such devices.

5.1.2 Transcoding

Teodosio explores combining and juxtaposing multiple forms of media in an electronic news environment [Teo91]. She attempts to determine multimedia primitives, particularly the non-lexical cues in audio and video news needed to effectively describe the content of news “articles” such as radio programs or television news. These salient features of the news articles are abstracted and stored in a database, and used to augment and link to stories in remote databases, transcode information across media, and display the news in different presentation styles.

5.1.3 Personalization

The author developed the personalization module. Before using Doppelgänger, it maintained a set of news preferences with strength and confidence weightings. A gradient descent algorithm was used to modify these in much the same way as Doppelgänger.
With the use of user models, articles are chosen so that the reader is presented with articles of which she may not have known she was interested in *prima facie*. This serendipity is one of the most compelling benefits of user modeling for electronic newspapers.

### 5.2 Data Structures

There are several data structures in Doppelgänger, each associated with different modules of the system. All are linked lists, and all fields in each data structure are stored as bytes. The adherence to this format was chosen with an eye toward implementing network-based communication between the modules, although the current means of communication is Network File Server (NFS) retrieval from ASCII files stored on disk.

The data structures are listed here in order from the most specific to the most general.

- **Sensations**

  *Sensations* are the raw data reported by the sensors monitoring users, one datum per sensation. *Examples*: *orwant at location 372, time 3578*, and *orwant at location 373, time 3591*.

  The sensor stores information in a sensation in Doppelgänger's *operator language*. It also marks the sensation with a timestamp, a strength rating, and a confidence rating.

- **Perceptions**

  *Perceptions* are prima facie inferences made from sensations. *Example*: *orwant move third-floor-corridor from north to south*.

  The observers, given the sensations produced by the sensors, generate perceptions. The types of prima facie inferences that are made use information specific to each sensor, and are hardcoded by the developer.

- **Inferences**

  *Inferences* are deductions or inductions made from integrating perceptions with a user model. *Induction example*: *Jon walks down the third floor corridor from north to south every day between 9 am and noon*. *Deduction example*: *Jon is coming into the Garden with probability 0.8. He will be logging in with probability 0.5*.

  Inferences are made by *deducers* and *inducers*, given the output of the observers and the current state of the user model.
• Actions

When not being accessed, user models reside on disk in a plaintext representation of these data structures. The deducer requires the entire user model to work, and the inducer requires the set of all user models.

Actions are exhaustive linked-list collections of inferences belonging to an "entity." Currently all entities must be users; the broad term is kept so that eventually inferences pertaining to collections of users can be supported.

• World

A World is a linked list of actions. Each entry contains all the actions for some user; the entire world data structure contains the actions for all users. The world contains the entire state of all user models used by Doppelgänger.

5.3 The Sensors

Each sensor-observer pair is executed as a UNIX process forked off the main Doppelgänger process. Thus, any combination of the sensors can be polled simultaneously, without interfering with the other operations of Doppelgänger, such as answering queries from client applications.

5.3.1 UNIX finger

The finger sensor periodically executes the UNIX finger command to decipher information about users. The login status, source of the connection, idle time, real name, office number, and office and home phone numbers are all parsed into a sensation. This is done by piping the result of the finger command through an awk script that extracts the different fields.

The UNIX utility last is not used because the polling frequency of the finger sensor is great enough that it is presumably not necessary—no one would log in twice in between polls.

5.3.2 UNIX aliases

Code was written to parse the aliases file on the Media Lab mail server. This file lists the mailing lists to which users subscribe. It is not complete; for instance, if users have accounts on other machines, and mailing lists on those machines, they will not be polled by this sensor. Since this file does not change often, it is polled daily.
5.3.3 UNIX lastcomm

The lastcomm sensor polls hourly, executing a lastcomm command for every user it knows about. The result is piped through an *awk* script, which extracts the command name. These are recorded as individual sensations.

The observer uses some UNIX knowledge to interpret these into more meaningful expressions.

5.3.4 UNIX calendar

The calendar sensor polls daily. It looks for a file named “calendar” in each user’s home directory. If found, this file is piped through an *awk* to extract the date and event name. The date is the converted to the Doppelgänger time format. The calendar entries are stored by date and keyword. Due to the infrequency of updates, the sensor is polled daily.

5.3.5 Badger

The Badger system, developed by the MIT Media Lab Speech Research Group, provides location data. Participants in the experiment wear badges that transmit an infrared bit pattern every fifteen seconds. Receivers mounted around the third floor of the Media Lab record these bit patterns and relay them to a central Sun Sparcstation II, which matches the bit pattern to a username. The Sparcstation makes this information available via Telnet, which is how the Doppelgänger badger sensor retrieves it.

5.3.6 Free Text

The first step in parsing free text is searching for metadata, i.e. information describing how to parse the text. It is assumed that metadata will precede all of the text, and will consist of a series of keyword-value pairs, separated by a colon. This style is followed by RFC-822, the Internet standard for electronic mail, as well as many other text formats. After being extracted, the metadata is used by the observer to classify the text: whether it is electronic mail, who wrote the document, and so on. If no metadata is available, the observer does nothing and passes the text on to the deducer, where rudimentary parsing and keyword identification takes place.
5.3.7 Newspace

Which articles the user reads, how long the user takes to read them, and which pages the user turns to are all sent as messages from the display application to Doppelgänger.

Abramson's network software uses TCP/IP to send these messages to "editor," the client application for Doppelgänger, which sends them via NFS to Doppelgänger [BLO+91].

In the future, Newspace will support a touch sensitive display. When this occurs, more sophisticated gesture recognition will become available to the electronic newspaper and therefore to Doppelgänger.

5.3.8 Model Manipulation

When a user types

```
doppelganger -v username
```

on the UNIX command line, the user's model is printed out, as long as either `username` belongs to the person typing the command, or `username`'s model is UNIX world-readable. The storage of the model imposes a structure on what is printed out: the most recent inferences are shown first.

Assuming they have permission, users may modify models with the `-m`, `-bs`, `-ds`, `-bc`, and `-dc` flags. The `-m` flag requires an object, attribute, strength, and confidence; these are added to the user model. The other four flags require an integer; they respectively bolster the strength, diminish the strength, bolster the confidence, and diminish the confidence, by the specified integer.

If an "edit" message is sent to Doppelgänger (`doppelganger -e username`), an emacs window is started, the model is loaded, and users can directly modify the values of the model. When the emacs window is closed, the model is saved out to disk, and Doppelgänger immediately starts to use the new model.

Heisenberg's Uncertainty Principle states that knowledge of an object's momentum is inversely proportional to knowledge about its position [Sha80]. Roughly speaking, this is because the act of observing an object changes it. This applies to Doppelgänger's model manipulation interface, since any act of viewing the model changes the model. This presents an interesting contention problem; it is solved, somewhat unsatisfactorily, by delaying the effects of these changes until after the user has finished viewing and modifying the file. This is implemented by locking the file which contains the user's model when the manipulation interface is activated. Only when the manipulation interface relinquishes the model are the changes added.
5.4 The Inference Engine

5.4.1 How Deducers Work

Once information has been processed by observers and stored in a perception, the deducer integrates the new datum into the user model. This may involve simply adding the datum, or substituting it for another, or ascribing meaning to it given information embedded in the user model.

The deducer uses two techniques for each datum it encounters. What each one does is decided by the operator language verbs of each perception. The deducer stores the data in the user model as object-attribute pairs, with associated strength and confidence values. Essentially, the objects and attributes are whatever the observer relays that aren’t recognized tokens of the operator language.

**Keyword indexing** builds and searches histograms of all incoming data. Any and all text available to Doppelgänger is searched for keywords. This is done on a NeXT 68040 computer using its *pword* program, which compiles word frequency lists indexed by “peculiarity” in English. This is especially useful for the more unconstrained data sources such as the free text sensor which produce difficult to parse output.

**Change maintenance** is used for the more constrained data indicating the current state of the user. If a perception reinforces information in the user model, the confidence is bolstered. If it conflicts with information in the model, the confidence is diminished, or the entry in the model superseded. The degree of change in the strength is inversely proportional to the old confidence value; a high confidence possesses inertia, while a low confidence can be easily swayed. Whether the datum is a reinforcement, corroboration, or replacement for other data in the model is determined by the operator language verbs and prepositions. What it does depends on the operator language verb. For instance, the verb *move* is hardcoded as being transitive, so if *A move to B* and *A move to C* are received, the first entry is superseded by the second. If the nouns of the operator language are the same, Doppelgänger combines perceptions and related inferences from the user model into a larger, more meaningful, datum.
5.4.2 How Inducers Work

When a client application requests an attribute for a user that is not mentioned in her user model, the inducer makes an educated guess by establishing correlations involving the attribute for the universe of user models.

If the user's model has the requested data, retrieve it. Otherwise:

1. Search the universe of user models for other users that have the requested data in their model.

2. If other models are found, for each different answer:
   - Calculate the similarity between it and the original model
   - Weight that model's answer by that amount

   The answer that has the highest similarity total is returned. The degree of similarity is reflected in the confidence rating.

3. If no other models share the attribute, return an "I don't know" message: strength 0, confidence 0.

This process has been tried for user populations of up to twelve users. While the performance was satisfactory, Doppelgänger will collect information about populations of hundreds of users in the future. When this occurs, inductions will take proportionally longer. The inducer's algorithm is \( O(n) \), i.e. linear.

A problem with this process occurs when attributes can be well-ordered, since there is no metric for calculating similarity between answers. Say some attribute can be expressed as an integer from 1 to 50. If 90% of the votes are cast for, say, values between 30 and 40, but the most votes were cast for 15, then 15 would win. It would have a low confidence value, but ideally a more representative answer could be supplied.

The inducer's algorithm is fast because information hidden in the strength and confidence values of the object-attribute pairs. However, if a user asks to see the reasoning behind an assertion using the \(-r\) (for reasoning) flag, an enormous amount of information is presented to the user, detailing every deduction and induction that led to the current assertion. Clearly, there needs to be some way to prune this list and separate the relevant inferences from the irrelevant. The inferences are presented to the user from the most recent to the least. This is a reasonable first approximation, but discounts inferences made a long time ago that have a large impact on the assertion.
5.5 Communication

Sensors speak only to observers, observers speak to both deducers and inducers. They communicate by passing around the data structures described above.

5.5.1 Operator Language

Inferences can only take place if the inferring modules have a grasp on the contents of the information relayed to them. This is accomplished by having the observers convert everything to a common operator language.

The operator language supports the following verbs: is, move, run walk, propel, grasp, ingest, state-change, information-transfer, give, speak, write, read, see, communicate, login, logout, and execute.

The operator language supports the following prepositions: near, with, at, to, from, and by.

The operator language facilitates the extensible architecture of Doppelgänger. New sensors can be easily added. Often, the information they have to relay can be expressed in the operator language. If not, the operator language can be extended; no modification of other modules is required.

5.5.2 Process Control

Doppelgänger runs continuously. Messages can be sent to it from the UNIX command line, commands such as "activate the badger sensor" or "stop tracking the electronic mail of Jon." When initiated, Doppelgänger forks one process for each sensor. This process contains the observer and periodically (the period depends on the sensor) writes data via NFS to a message bin.

Every minute, Doppelgänger polls the message bin to see if any new messages have arrived. If so, Doppelgänger collects the messages and sends them to the deducer module, which forwards them to the inducer module. Every hour the entire state of the system is saved to disk as a precaution.
5.5.3 Server-Client Communication

Client applications request and retrieve personal information from Doppelgänger. To these applications, Doppelgänger looks like a server. This server-client communication is primitive; all information is passed through UNIX standard input and output.

Queries are sent to standard input of the workstation Doppelgänger is running on using the "-q" flag. Queries can be phrased in either the operator language or in simplistic English.

Every command has the same effect; a string is sent to standard output. The string has four numbers- the strength of the assertion (how much Jon likes topology) with the maximum possible strength, and the confidence that the strength value is correct with the maximum possible confidence.

5.6 Hardware Platform

Doppelgänger consists of 50 kilobytes of C. Development took place on DECstation 5000 and 3100 workstations running Ultrix 4.0. It is an ongoing project.
Chapter 6

Issues

This section discusses some of the issues raised by powerful user modeling systems such as Doppelgänger. They are meant to provoke thought, not to provide final answers.

6.1 Privacy

User modeling systems as ambitious as Doppelgänger contain the potential for great threats to privacy. In 1991, Lotus began to market a product called the Lotus Marketplace, which contained personal information about twenty-one million individuals, including their address, occupation, and estimated income based on their residential neighborhood. Outrage over this invasion of privacy from a small but angry and vociferous contingent was so great that Lotus withdrew the product. Yet Lotus contained nothing more personal or speculative than an estimate of household income garnered from a neighborhood average. The direct marketing industry thrives on buying and selling such information. If systems such as Doppelgänger were commonplace, the dangers would be made an order of magnitude greater.

Doppelgänger is constrained to the Media Lab; by itself it poses no threat to the privacy of individuals. All participants in this research experiment are volunteers. Even so, the notion of being “watched” by several unknown, mysterious “sensors” is not a pleasing one for most; witness the anxiety of secretaries whose keystrokes are monitored so that their supervisors can monitor their performance. Such a task is easy for Doppelgänger.

6.1.1 Big Brother

If a user model is to be useful for any but the simplest tasks, it must contain a plethora of information about the user. The act of inferring information about the user by necessity
involves collecting information that the user may want kept confidential. Some consider idle time at a computer console or the identities of electronic mail senders too personal for public consumption. When user models reach their pinnacle, tracking knowledge, beliefs, goals, plans, schedules, behaviors, abilities, preferences, and misconceptions, there will be tremendous potential for exploitation of the user. However, it is important to realize that such risks are not necessarily concomitant with personalization.

Of paramount importance is that private information not be released to other entities without the user's consent. Once such information is released, it can never be returned to the user's sole possession; information is an elusive type of property, but one that will become increasingly more precious as we live more of our lives in a computational environment.

The user should always be able to ascertain what the computer knows about him. Every datum inferred should have an associated confidence level, and the user should be permitted to change anything with a confidence below a certain threshold. Or, the user should be allowed to change anything he wants, but information so modified is tagged so that generalizations across users do not make a wrong conclusion.

If all applications of user modeling are considered, it becomes apparent that not all information should be under direct control by the user. As an example of data that shouldn't be modified, consider the maintenance of consumer purchase and credit information for a credit agency. If people could, they might modify this data to their advantage.

6.1.2 Assurances

There are two layers of protection in Doppelgänger: First, users can hide from the sensors. Second, they can protect all or part of their models.

Doppelgänger is a localized system. You can always turn it off, and users always know when it's working. Doppelgänger allows humans to observe precisely who is being monitored, and which sensors are being used. Its reasoning is always available, and the justification for its educated guesses is always presented.

Users can prevent their being seen through the eyes of any sensor, or can hide altogether, by sending a simple message to Doppelgänger. Mistakes can be identified and corrected because the information Doppelgänger has on a person is always available and modifiable.

Protection of user models can be accomplished by encoding the information or otherwise preventing it from being seen by undesired machines or people.
6.2 Doppelgänger As An Aid To Introspection

Users can learn a lot about themselves by perusing their models in Doppelgänger. Formally, Doppelgänger’s purpose as a server is to answer questions that people or applications have about users. But Doppelgänger can also provide users with interesting questions to ask themselves by elucidating patterns of which users were not previously aware. Consider the Newspace application. If either Doppelgänger or Newspace knows about a certain style of writing, it might notice that a user consistently prefers articles of that style. This will be apparent in the user model, and the user might learn for the first time that there was a name for that type of article she likes so much.

6.3 Models of Models

As models evolve in sophistication, three interesting ideas should be considered: how the user models himself, how the user models Doppelgänger, and how Doppelgänger models Doppelgänger. An interesting recursion can occur when Doppelgänger considers the user’s model of the user’s model of the user’s model...

6.4 Intrusiveness and Confidence

Systems such as Doppelgänger and Newspace must gain the user’s trust for success. If the user doesn’t have confidence in the system’s ability to make these choices for him, the technology will be rejected.

A quick way to lose this trust is through a system that intrudes on the user. User modeling should be unintrusive but not invisible. The more users are reminded of the existence of user modeling systems and the fallible inference mechanisms and sensors behind them, the more nervous they will become.

Better than any feedback from client applications is the ability to ask the user questions to ensure that the user model is being constructed accurately. When should user modeling systems ask the user questions? Carberry’s process models initiate “subdialogs” with users when contradictions need to be resolved. However, if the user is asked at inopportune times, or just too often, she will quickly tire of being harangued by the computer.

Kass and Finin raise the question: who has the responsibility for ensuring that the communication is successful, the user or the user modeling system [KF88]? User modeling systems are by necessity complex; the user should not be expected to understand all the
inference mechanisms, faults of the sensors, and all the nuances of the user model, since most of it is boring, and all of it is changing. To ensure acceptability, user modeling systems must bear the responsibility for successful communication.

A general idea of how the system works should be enough. Decades from now, when user modeling systems are really good enough that they can be trusted, they can slip beneath the user's consciousness completely.

6.5 Common Sense

When inference mechanisms make assumptions about human behavior, they need common sense. In this thesis, this common sense is incorporated directly from the observer developers into the code. Ideally, user modeling system could draw on a common sense database that would shed light on the reasoning behind people's workaday actions, and use that information to make more powerful inferences about information received. Ongoing work in the Media Laboratory Music and Cognition Group attempts to understand stories based on principles from Minsky's Society of Mind [Min86]. This effort treats stories much the same as Doppelgänger treats people: the tricky inferences are not represented explicitly but rather emerge from a large mass of raw and semi-raw information. This is called episodic reasoning.

6.6 Other Sensors

What types of sensors will Doppelgänger be able to use in the future? Doppelgänger uses eight sensors, of which the four UNIX sensors are similar. Future sensors promise an even broader range of modes of input about the user. The UNIX computational environment has additional sensors available now, and will surely support more in the future.

An eye tracker could be used for a lot more than what Crosby and Stelovsky [CS90] use it for. They can be used not only to identify where on a computer screen a person is looking, but also where in the room the person is looking or who she is talking to. Unfortunately, it is cumbersome, but lighter head-mounted displays are being developed as well as trackers not mounted on the head at all, in the MIT Media Lab Advanced Human Interface Group. Also being worked on in this group is the dataglove, which records hand positions [Bol84]. The dataglove affords gesture recognition, making "body language" and kinesthetics available to user modeling systems. Like the eye tracker, the dataglove is overly cumbersome.
Vision offers both short- and long-term sensors that Doppelgänger will be able to use. Clearly, true machine vision will offer incomparable benefits, but less ambitious projects that are feasible now can also be useful. A system that could identify faces would be useful for user identification; “Eigenfaces” could make this possible [TP91]. Cameras that record depth information and use it to build models of a scene could also prove useful [Bov89].

Most user modeling researchers address conversational dialog, but none consider actually using speech. This is presumably because formal speech recognition is difficult, but as work at the MIT Media Lab Speech Research Group shows, full recognition is not necessary for speech to be useful for a user model. Affect in spoken discourse can be used to infer spoken “mood” [Cah90]. A system in which the user’s speech is continuously recorded, and periodically marked by the user as being worthy of saving [Hin91] would be very useful for a user modeling system.

Speech recognition, natural language processing, and machine vision are all grails on which many researchers are working feverishly. These goals are a long way off, but will make new levels of user understanding available to Doppelgänger when they are attained.
Chapter 7

Conclusion

Doppelgänger makes the following contributions to the field of user modeling:

Sensors:

- Full exploitation of the computational environment in real time
- Use of non-dialog, unintrusive sensors
- Providing an *extensible architecture* for adding sensors
- Integration of multiple sensors

Models:

- Stereotypes, sensors, and models are dynamic
- An abolition of the "session" in user modeling through continuous data gathering
- Use of other's models, as well as interpolations between models and "caricatures"
- Maintenance of real-time ephemeral user traits (such as idle time)

Knowledge:

- Automatic generation of implicit stereotypes
- Providing an interface through which users can view and modify their models and the reasoning behind them
- The user provides as little or as much intervention as desired

Human factors research is similar to user modeling in that both attempt to construct a model of people. Human factors, however, has concentrated on a general model of people: identifying common traits among people that can be exploited by a computer system. In contrast, user modeling seeks to identify and exploit differences between people.
All interactive computer applications can benefit in some way from increased knowledge of who is using the application. Doppelgänger, through its monitoring of varied sensors in different media, becomes better able to deal with people as it learns about a population of users. The longer it runs, the more it knows, the more it can infer, and the more detailed and accurate the information it can supply to clients.

7.1 Extensions to Doppelgänger

The ability to add new, unanticipated sensors is intrinsic to the design of Doppelgänger; such extensions are expected. This section discusses more fundamental improvements to Doppelgänger that are worthy of consideration.

7.1.1 The Interface

The templates for sensor and observer development aren't as simple as desired. This is in part due to the UNIX/C development environment of Doppelgänger; LISP would facilitate development, but is less well integrated into the UNIX computational environment. The observer code is obtuse; developers cannot immediately understand how it works. The sensor code is obtuse also, but this is justified because Doppelgänger can't possibly anticipate the wide variety of sensors. There should be an automatic way of producing more than just an observer template: users should be able specify rules, and have them compile automatically into an observer.

A graphical interface would be an interesting addition to Doppelgänger. It is not as important as Kobsa contends [Kob91], because development in Doppelgänger is meant to be a less intricate process than in BGP-MS. Still, it could be a useful visualization tool for identifying patterns, rules, and stereotypes.

An unresolved issue is how and when Doppelgänger should query the user. Currently, it doesn't at all, which is a drawback. The user can explicitly modify her model, but she can't be expected to maintain a constant vigil. Doppelgänger should be able to ascertain the receptivity of the user to being bothered by questions, and weigh that against how desperately Doppelgänger needs its information before bothering the user with a questionnaire.

56
7.1.2 Performance

Doppelgänger has the potential to be unbearably slow because of its implicit generation of stereotypes. It is fast enough for its purposes now, but may not be when the numbers of sensors and users increase by orders of magnitude. In her system for autonomous agents, Maes gets around this problem by trading thoughtfulness for speed when necessary. This compromise would be a good one for Doppelgänger to use when dealing with real-time applications such as Newspace. An application would be able to say, "I need an answer to this question within the next second," prompting Doppelgänger to make the most educated guess it can given the time constraint.

Another problem is the accumulation of worthless information, both because of strains on disk space and on the user—showing the entire chain of reasoning used to make an inference becomes exponentially unwieldy when the number of causal assertions is high. Much information is discarded when Doppelgänger can identify some datum as superseding previous information, but this is not and will never be possible all the time. Doppelgänger needs a method of identifying and pruning unimportant information.

A natural feature to be added to Doppelgänger is a better message-passing system than NFS. This could be done using Abramson's network code [BLO+91]. This mode of communication could also be used for communication between Doppelgänger and clients on the local network, while external machines could submit their queries by electronic mail.

7.1.3 Knowledge and Inference Mechanisms

A "common-sense" database or a world-knowledge database would be a valuable addition to Doppelgänger. Such explicit representations are spurned in this thesis not because they are intrinsically a bad idea, but because these things are very hard to do properly; doing so is beyond the scope of this thesis.

Doppelgänger needs a better parser. This is especially apparent when free text is parsed; most of the text is thrown away because the sentence structures are too difficult for Doppelgänger's parser. The text ends up being useful primarily because of the histogram of peculiar words, through which recurrent concepts can be identified.

One curious characteristic of Doppelgänger is that it requires outside feedback to learn; if left alone, it will merrily go along making deductions, but never receive a clue as to how it is performing. The less Doppelgänger interacts with the world, the less it learns about how well it's learning. A self-questioning mechanism might be useful to ensure consistency
of Doppelgänger's beliefs: the system could serve as its own application, with feedback provided through the user-querying interface described earlier.

Another desirable feature to add would be the ability to query about a population, e.g. "How many people at the Media Lab ride a bicycle to work", or "Do male students like computers?"

7.2 Future Work in User Modeling

What's next for user modeling? Hopefully, Doppelgänger will shake up the user modeling community enough to make it adopt some of the ideas introduced in this thesis. The differences between Doppelgänger and other user modeling systems underscores the need for more flexibility in their use. Unfortunately, this flexibility is attained only at the expense of ease of design.

The sensors available to user modeling systems will increase as people realize the benefits of personalization, and the value of inferences obtained from seemingly uninformative sources. Every datum helps. As people spend more time in a computational environment, a wider range of activities will become available to these sensors. In the future, user modeling systems will hopefully become not merely parts of separate applications, but integrated into entire computational environments, serving out their data to a new generation of applications that customize themselves for the user—no menus needed.

The most significant deficit of current user modeling systems, which Doppelgänger avoids, is the restriction of input to keyboard discourse. There are as many ways to convey information to a computer as there are to humans; typed dialog is only one, and a poor one at that. If no other modes of communication are used, user modeling will suffer a fatal handicap.
Acknowledgements

Walter Bender has provided constant support as my advisor throughout this thesis. His supervision provided just the right amount of direction while allowing me to fly off on my own tangents. Walter combines enthusiastic ambition with a mellow attitude, and there is little more one can ask for in an advisor.

A hearty thanks to the inhabitants of the Media Lab Garden for making my work environment an enjoyable one.

Thanks to the MIT UROP program for supporting undergraduate research. It is the reason I came to MIT, and my UROPs at the Artificial Intelligence Laboratory and Media Laboratory have been the most valuable experiences of my career.

Thanks to IBM for their financial support of my work.

My Media Lab education does not culminate with this thesis; as I embark upon graduate school under Walter, I look forward to more of the enjoyable times I have experienced as an undergraduate.
Bibliography


[umu] User modeling and user-adapted interaction: An international journal. quarterly.


