i-Seek: An Intelligent System for Eliciting and Explaining Knowledge

BY
Ashwani Kumar

BACHELOR OF TECHNOLOGY IN ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR, INDIA 2001

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2005

SIGNATURE OF AUTHOR:
PROGRAM IN MEDIA ARTS AND SCIENCES
MAY 12, 2005

CERTIFIED BY:
THESIS SUPERVISOR
HENRY LIEBERMAN
RESEARCH SCIENTIST, MIT MEDIA LAB

ACCEPTED BY:
CHAIRPERSON, DEPARTMENT COMMITTEE ON GRADUATE STUDIES
ANDREW B. LIPPMAN
PROGRAM OF MEDIA ARTS AND SCIENCES

i-Seek: An Intelligent System for Eliciting and Explaining Knowledge

B Y

Ashwani Kumar

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning on May 6, 2005
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences

ABSTRACT

We propose *i-Seek*, an Intelligent System for Eliciting and Explaining Knowledge that leverages the OpenMind [1] Commonsense knowledgebase in conjunction with domain-specific knowledge in Personal Finance, Technical Help, and Health domains to act as an advisory system for novice users. Most of the interfaces are plagued by recurrent key problems: 1) *elicitation* – how to ask questions that enable the expert model to make decisions, and at the same time, are understandable to the novice, and 2) *explanation* – how to explain rationale behind expert decisions in terms that the user can understand. *i-Seek* maps the user's goals and expectations to the corresponding expert model's attributes as expressed in domain-specific terms. For example, instead of asking "What is your risk tolerance?", where the user might not comprehend the notion of risk tolerance, *i-Seek* tries to *elicit* the same information by asking a non-direct question such as "Do you usually buy lots of lottery tickets?". *i-Seek* constructs the novice user model by taking into account the user's personal information, interactions history, and the current context.

Thesis Supervisor: Henry Lieberman
Research Scientist, Media Arts and Sciences

Acknowledgments

It has been a great pleasure and a hugely rewarding experience working with Henry Lieberman, my research advisor at MIT. First and fore mostly, I would like to thank him for all his support, belief, and help that he provided no matter how tough the situations were. Henry always provided his valuable guidance for this work and beyond. I also want to thank him for all the proactive supervision that he provided for this thesis, without which this work would not have been possible. I would also like to thank the members of Software Agents and Commonsense Computing groups, especially Push Singh, Ian Eslick, Hugo Liu, Alex Faaborg, and Jose Espinosa for helpful discussions and advice. Special thanks to my thesis readers Dr. Dan Ariely and Dr. Raj Goyal for not only being my thesis readers but also showing active interest in my research. I want to thank Pat, Linda and Mary for their unwavering support and understanding on administrative matters. I have also benefited immensely from the larger MIT and Media Lab community to which I would like to thank collectively for making life at MIT fun and stimulating. I am thankful to our collaborators at AOL, Amy Hale, Tom Jarmolowski, and Gail for sponsoring me as a fellow and taking active part in the project.

Special thanks to my fiancée, Erica for her constant emotional support and love, which has always kept me focused and on the right path. Thank you for being such a great friend and partner. Last but not the least, I would like to thank my parents, sister, and brother for their love, support, and sacrifices they had to make for my sake.

Table of Contents

Figures

Tables

1 Introduction

1.1 Problem

Computer systems with common sense have been a persistent dream of A.I. for more than 50 years. Early research showed that some factual knowledge about the real world could improve the system's performance and accuracy significantly [2]. However, most of the prevalent expert systems and interfaces lack this kind of knowledge and hence, are very difficult to use by the novice users.

Figure 1: Dilbert's joke about experts

The same is true about most of the existing expert advisory systems as is mocked by Dilbert in the above cartoon (Figure 1). These systems employ inflexible interfaces like menus, drop-downs, buttons etc. (see Figure 2 and 3 on the next page) providing only a handful of options that are sometimes not immediately clear or intuitive and may potentially collect superfluous information. For instance, instead of asking the user how risk tolerant (s)he is by offering some discrete and incomprehensible levels of risk (Figure 2), it will be enormously beneficial to provide some explanation about the ramifications of being aggressive or taking high risks [3]. Common Sense facts such as "frequent gambling

means high risk" and "having recurring credit card debts means high risk" capture human tendencies of risk taking attitudes.

Risk Tolerance

How you feel about risk affects how aggressively or conservatively you should invest. The chart below shows how much an investment of $10,000 may fluctuate during a one-year period, depending on how the amount is invested. However, the maximum gain or loss on an investment is impossible to predict. The ranges shown in the chart are hypothetical and are designed solely to gauge an investor's risk tolerance.

Given the potential gain or loss in any one year, in which would you invest?

○ **A** (loss of $164, gain of $593)

○ **B** (loss of $1,020, gain of $1,921)

○ **C** (loss of $3,639, gain of $4,229)
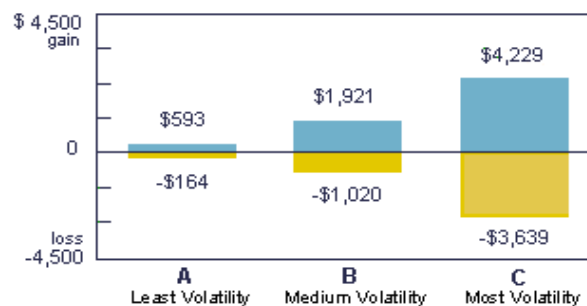
Figure 2: Vanguard's Risk Tolerance Question

http://www.supportwizard.com/sw/bin/17/supportwizard.cgi

Getting Started   Latest Headlines

Find Answers

Search | Power Search

1. **Choose The Topic:**
   Technical Questions ▾

2. **Narrow Your Search Using Keywords:**

   Combine multiple words as: ○ Or ◉ And ○ Exact Phrase

3. **Press the Button:**
   GO

Login and Ask Us Go Back

Status: No results found

★ **Powered by**
   SupportWizard

Figure 3: A Sample Technical Help Page

Even more importantly, these interfaces have unwieldy and unnatural ways of estimating diagnostic objectives for the domains. For example, in the following WhatHaveIGot Headache survey questionnaire (Figure 4), the objectives are divided into enumerated categories like acute vs. chronic, which certainly make the interface development easy but at the expense of usability of these interfaces. The interface presents the user with a large number of choices to quickly cut down the search space, but that leaves the user tediously perusing large numbers of irrelevant choices.
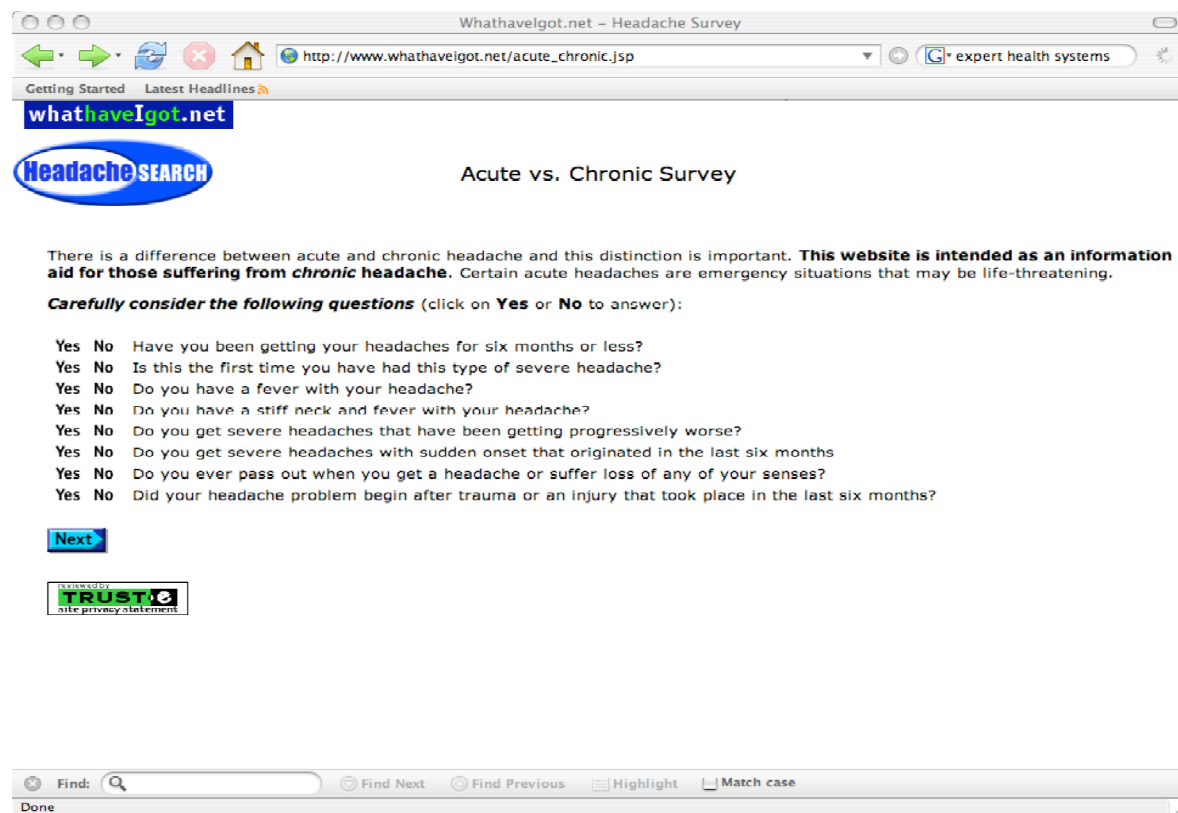
Figure 4: WhatHaveIGot Headache Survey

The proposed *i-Seek* system strives to overcome these limitations by adopting a commonsense-enabled analogical reasoning approach [4,5,6] for mapping novice and expert knowledge, without compromising on the value of expert domain model. The systems uses common sense reasoning to facilitate seamless interactions between the user and the system by providing intuitive ways of bridging the knowledge gap [7]. We illustrate the generality of the i-Seek architecture in the domains of Personal Finance and Technical Help. Figure 5 on the next page shows some sample OpenMind knowledge for the utterance, "invest money". *i-Seek* uses knowledge pieces like those to construct adequate *elicitation* and *explanation* models as described in subsequent sections. The *i-Seek* design enables it to be extensible, scalable, and modular.

1.2   Background

In 2000, the Commonsense Computing and the Software Agents groups at the M.I.T Media Lab launched the Open Mind Common Sense (OMCS) initiative. The primary motivation behind the project was to aggregate common sense knowledge in form of English sentences from the WWW users [8]. The project is a great success and thus developed OMCS corpus now contains over 700,000 facts about everyday human life.

Since then, the OpenMind project has led to several interesting projects. Some of them use the corpus to develop innovative interactive applications, while others use the knowledge base to enable various types of reasoning frameworks [7,9,10,11,12]. One of the interesting experiments out of these projects was to evaluate the domain-specific coverage of OpenMind and in what ways can this knowledge be used to build collaborative interactive systems that use or are at least behaviorally similar to some sort of expert system.

Figure 5: Open Mind Sample Knowledge

We felt that there was a great need for an i-Seek like system architecture in many industrial applications. Increasingly, as most of the Internet services are being commoditized, the key differentiation for the companies is providing better customer service and support. Specifically, AOL has built its business around enabling its solutions and customer service to hand hold end-users and solve their problems concerning Internet connectivity and computers in general. AOL extensively pursues research to further enhance their capability in providing help and assistance to people who are novice users.

In the similar vein, AOL has generously provided support for our research and has been actively involved in design and testing of the system. Tom Jarmolowski and the Help team at AOL have worked directly with us in providing useful insights, data, and technical help. The designing and implementation of the system has been an iterative process and we have always benefited hugely from AOL's feedback and critiques about user scenarios, domain-knowledge coverage, and efficacy of the system. In the process, we asked AOL to provide some examples of scenarios where a user's problem can be solved by providing an analogy from commonsense knowledge. In the following are two sample knowledge pieces from the data provided by AOL:

1) "Use the System Information Tool to delete the AOL adapter and restart the software

Explanation The AOL adapter transmits information from the Internet to our service. When this adapter is broken, data can't get through so you will not be able to get to Websites. When the software opens, it checks to make sure necessary files are installed. If the AOL adapter is missing, it will install a new adapter. **NOTE:** The WAN Miniport adapter replaces the AOL adapter for newer versions of the software.

Analogy Think of the AOL adapter as a bridge and Members as passengers on a train. If the bridge is broken, the people can't get to their destination. Once the bridge is repaired, the train can resume its journey and reach its destination."

2) "Check if cipher strength is '0' Upgrade Browser to 128 bit Encryption

Explanation Cipher strength or encryption refers to the built -in security features of your browser. Generally, Websites require 128-bit encryption in order to process information securely. If the cipher strength of your browser is inadequate, you will not get into secure Websites. Upgrading your browser's encryption may help it better handle secure Websites.

NOTE: You only need to do this when unable to get to secure Websites.

AnalogyIf you don't have the proper security clearance, you may be able to get into the building, but not into certain areas. You must upgrade your security clearance status to go further. So without the proper encryption, your browser may be able to access a website, but not log in."

Based on these encouraging results, we developed applications based on the i-Seek architecture to help automate the process of selecting analogies in order to improve the effectiveness of help advice.

1.3 Organization

This thesis discusses all work done to date on the *i-Seek* project and lays foundation for future work in order to extend the framework in other domains. Section 2 discusses some user scenarios and builds the case for commonsense reasoning. In Section 3, I describe the functional architecture of the *i-Seek* system. Section 4 describes the commonsense-enabled analogical mapping, which helps in building *elicitation* and *explanation* models for domain-specific applications. Section 5 describes some of implementation, while Section 6 discusses the user evaluations. Also, I provide an account of contemporary research in this area in Section 7. Finally, we discuss the outcomes of this project in Section 8.

2   User Scenarios

In the following sub-sections we provide illustrations of domain-specific interactions between the user and *i-Seek* in two application domains of Online HelpDesk, and Personal Finance. *i-Seek* leverages Commonsense Reasoning to map expert finance and help-related information to general personal life situation and vice versa.

2.1   Online HelpDesk Scenario

John is an AOL user and he has been using its service for past 2 months. He uses it to surf Internet, look at his friend's pictures online, and send emails. Recently, he is discovering that his Internet Explorer is getting slower and slower day-by-day. He wants to know what is causing this and how to remedy it.

Help Topic: "Browser is running slow"

He invokes the AOL Commonsense SuggestDesk interface to Suzy, the human chat assistant sitting behind the interface (Figure 6). SuggestDesk doesn't directly interact with the user, but instead provides relevant suggestions in form of analogies, diagnostic information about the user problem, and explanations so that Suzy can use them to provide well-informed advice to the user. The goal is not to replace the human assistant, but to make the help process as helpful as possible.
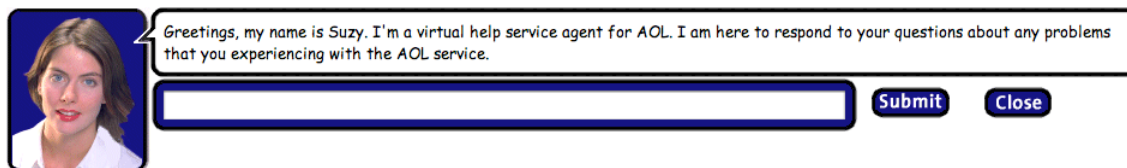
Figure 6: Mock-up of HelpDesk Assistant

John can use the interface to ask questions about his problem. He types:

"My Internet Explorer is running slow"

[Queries can be in form of keywords or simple English sentences.]

The HelpAssistant builds a simple parse of the query and processes as follows through CommonsenseAnalyzer:

a) It searches OpenMind for knowledge relevant for browsers such as following:

Internet Explorer is a web browser

surfing the web requires a web browser

if you want to surf the web then you should connect the computer to the internet and use your browser

Your web browser accesses the Internet so that you can view webpages

a web browser is for viewing web pages.

my browser can view images.

webpages contain text and images

b) From the knowledge acquired from OpenMind, the analyzer builds a frame-based representation of the user's parsed query with slots as relations from ConceptNet:

[Internet Explorer] - > used for surfing the web

[Internet Explorer] -> connects to internet

[Internet Explorer] -> can be used to view [webpages] - which contain text and images

c) It also queries the expert help model to fetch domain-specific knowledge about browsers and in which situations they would run slow:

Browsers download pictures and files to computer.

It helps in faster reload of the webpages.

These files are known as browser cache or Temporary Internet Files.

After sometime, the cache size may build up and cause the browser to slow down if it is not cleaned.

Browsers can be vulnerable to viruses. some free applications can have viruses.

Viruses use browser's resources. This may cause the browser to run slowly.

Deductions:

[Internet Explorer] -> can have large cache after months of use

[Internet Explorer] -> can run slowly if the cache is large

[Internet Explorer] -> can run slowly if infect by virus

Elicitation Step:

The system provides the human help assistant with analogies and possible diagnoses with respect to problem at hand and analogical concepts. As the assistant now knows that there exist multiple potential causes for the problem, she tries to elicit more information from the user (see Figure 8 for sample elicitations):

Suzy: "What do you mostly use your AOL service for? Some services could be, downloading free software, looking for friends photos, or just surfing the Internet."

John: "I never download free software but yes, use it for emails and looking at my friend's photos."

Suzy can now infer that as security is not the issue, perhaps browser cache is the problem at hand.

[Internet Explorer] - > used for surfing the web

[Internet Explorer] -> connects to Internet

[Internet Explorer] -> can be used to view [webpages] - which contain text and images

[Internet Explorer] -> can have large cache after months of use

[Internet Explorer] -> can run slowly if the cache is large

She tries to confirm that in the next step.

Elicitation-Support Step

Suzy: "How long you have been using the service?"

John: "about 2 months"

[Internet Explorer] -> has large cache after months of use and if it is not cleaned.

[Internet Explorer] -> run slowly if the cache is large

Suzy deducts using the above information that the browser performance issue is most likely because of a large cache, which has not been cleaned. The system provides her similar situations in expert's knowledge base and finds the following steps:

a) Go to AOL menu and select Preferences tab.

b) In the displayed window, select WWW

c) You should see an "empty cache now" button.

d) Click it!

Explanations Step:

Finally, the system translates the expert knowledge into novice's language and finds an analogical situation from Open Mind.

Suzy: It looks like the browser cache on your computer is too large and is causing the Internet Explorer to run slowly.

Reason: When you visit Websites, they download pictures and files to your computer in order to display properly. These files are known as browser cache or Temporary Internet Files. Eventually these files may become damaged and cause pages to load incorrectly, or they may just build up and need to be cleared out.

Solution: So, you should clear the browser cache. It is like when your trashcan is full, and you can't throw in more garbage. Clearing the browser cache is like emptying the trash. You have a folder full of files you no longer want or need. It's time to empty it.
Here are the steps:

a) As you will first locate the trashcan in the house, you should first locate the AOL tab in the interface and locate Preferences.

b) As you will pick the filled trashcan, you should pick the icon that is of browser marked WWW.

c) Finally, as you empty the trashcan, you should click the "empty cache now" button.

d) Restart the browser and it should run faster now.

2.2 Personal Finance Scenario

John has no prior investing experience and wants to start investing small amounts. He is seeking some advice about it.

Dialogue:

User: I have some money/ I am new to investing/ I want to start investing/ I do not have much money to invest

System reasons about what to do with money with respect to the assumptions and constraints:

CSReasoning:

      cause-of ("earn money", "invest")

      isa ("activity", "invest money")

      effect-of ("earn interest", "invest money")

      effect-of ("financial return", "invest money")

      cause-of("buy real estate", "invest money")

      effect-of("financial return", "sell real estate")

      following-event ("sell", "buy")

Commonsense Knowledge:

      money is earned through working or investing

      An activity someone can do is invest money.

      If you want to fight inflation then you should invest your money in high yield accounts

      dollars are a form of money, which can be invested to earn interest

Something that might happen while investing money is loss

investing money is for keeping away from boredom

investing money is for showing support for a cause

Something that might happen as a consequence of investing money is becoming attached to the outcome

You would invest money because you want to save it

The effect of opening a business is investing money in a business

a worthwhile cause would make you want to invest money

You would invest money because you want return

Expert Reasoning:

if desired-effect("earn money", X) && isa ("activity", X) && X: parameters("new", "not much money", "beginner" etc)
Then,
John's profile set as beginner and associated defaults such as small investments, moderate risks, appropriate diversification etc.

System:
Should not wait while your cash flow improves.. There are ways for beginners to invest.

Why?
Explanation: If you invest early you can sell your investment later for high financial returns

Analogy: If you buy real estate now, you can sell it after for high financial returns.

Expert Reasoning:
if invest-state("beginner", X) && X:parameters ("small investment", "moderate risk", "mutual funds" etc)

a) Direct Investing: method of buying stock directly from the company without going through a broker

Expert Knowledge:

    Use direct investing if you want to save on broker fees

    With direct investing fees are lower than of broker fees

    you can invest small amounts (often as low as $25 or the value of one share of stock)

Analogy: buy/rent house directly from the house owner

Commonsense Knowledge:

    a real estate broker would charge you money

    if you rent house directly from house owner you will save broker fees

    to save money do not use real estate agents

or

b) enroll in programs to have fixed amount deducted from your bank A/C and automatically invested in stock

Expert Knowledge:

    stock purchase can be done through bank account

    DPPs allow automatic investment

Analogy:

It is like setting up automatic electricity bill payment.

Commonsense Knowledge:

    bills should be paid on time

    you can setup your bank account for automatic bill payments

    You can pay electricity bills automatically through your bank account

you can pay heating bills automatically through your bank account

User: Great! I will start investing right away.

Assistant: Please look at the following sites that provide DPP plans:

www.dppinfo.com

www.moreaboutdpp.com

In this section, we described one Personal Finance and one Technical Help scenario, which illustrate the system's reasoning process in mapping expert knowledge to the novice knowledge. In subsequent sections, we go more deeply into these scenarios and illustrate how the architecture implements commonsense reasoning to be able to perform these analogies. Also, in the Discussions and Future Directions section, we talk about a Personal Health scenario, which will further demonstrate the generality of our approach.

3 Architecture Description

3.1 Commonsense Reasoning Components

OMCS is a corpus with 700,000 facts about everyday human life. Using NLP (Natural Language Processing) techniques, the Commonsense Computing and Software Agents Groups have created a variety of toolkits to enable applications with common sense knowledge:

(a) ConceptNet [15] is a semantic network extracted from OMCS. It has semantic relations like (CapableOf "person" "keep dog as pet"). With this kind of operation the system is able to infer the context given a situation. For example, the context of "car" is "wheel", "travel", "street", "drive", "container", "bridge", "drive", "parking lot", "automobile."

(b) LifeNet [16] is a dynamic bayesian network, which mimics an egocentric model of human daily life. It has links like (=> 0.858 "I enjoy music" "I watch musician perform"), where 0.858 is the probability that this link is true.

(c) ExpertNet has been developed as part of this project to enable a commonsense inference tool for reasoning about expert knowledge situations. The expert knowledge has notion of "Situations". Situations are cluster of propositions with some joint probability distribution. The semantic network models individual situations as graph nodes and predicative relations between any two situations as graph edges. Some of the reasoning capabilities of the net would be to do nearest neighbor search, exclusion of some propositions from a situation, and to infer likely situations given current situation etc [11, 12]. For instance, given a Situation s = {"I want to buy a house", AND "I do not have much money"}, another Situation s1 = {"I want to buy a house", AND "I look for houses in an inexpensive neighborhood", AND "I contact real estate agent", AND "I apply for mortgage"} would be likely inferred as the correlated neighboring situation. ExpertNet

helps i-Seek in modeling domain information in a semi-formal manner and is used to provide knowledge resource which can be reasoned about using Commonsense reasoning.
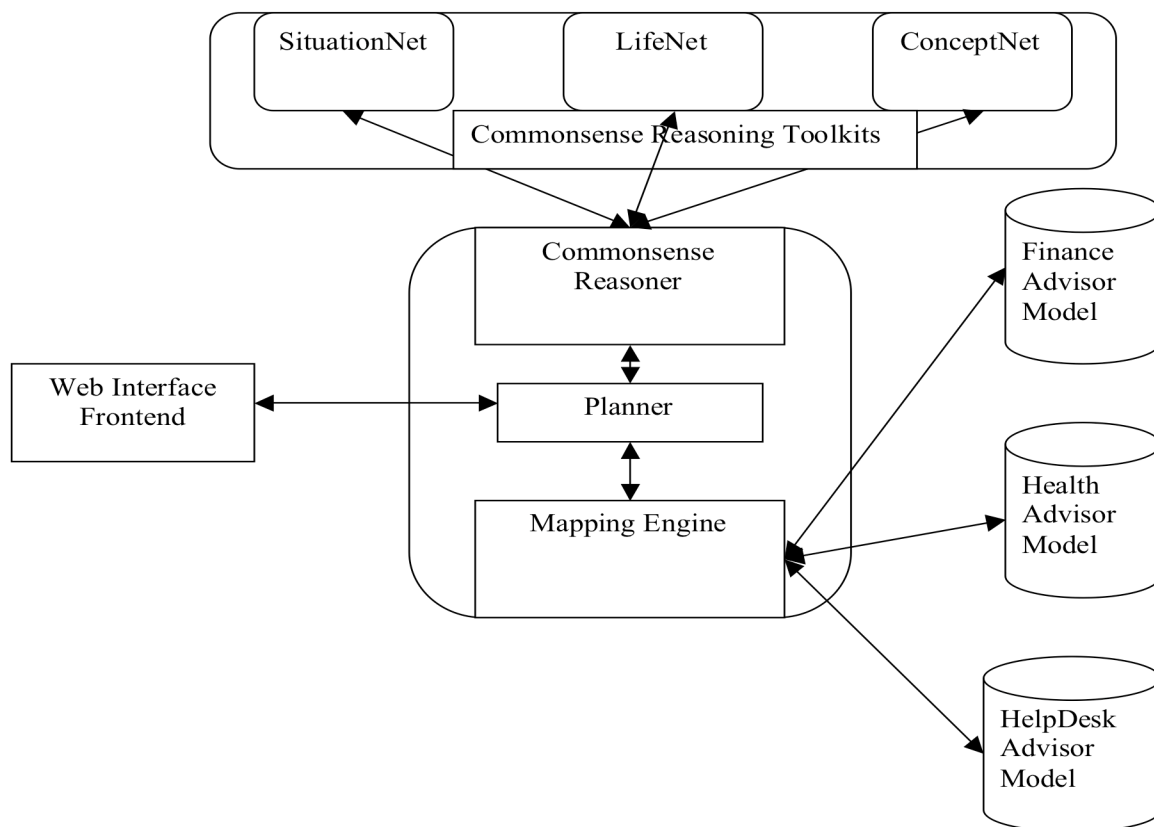
Figure 7: *i-Seek* Functional Architecture

3.2 Domain Models

i-Seek has 2 expert domains – 1) Personal Finance, and 2) Online Technical Help. Accordingly, *i-Seek* maintains a Personal Finance Advisory knowledge base, and an AOL-specific online SuggestDesk knowledge base. OMCS may have sparse knowledge in the domain of interest, such as finance. Thought it might have general knowledge such as "investing in stocks is risky", it might not have all such relevant facts. Therefore, we anticipate the need for experts to augment the Common Sense knowledge with more specific facts, such as "high beta-ratio usually means riskier stock". Even this could be considered "Common sense" for the community of financial advisors. This more specific knowledge can be handled in a manner similar to the more general Common Sense.

3.3 Reasoning and Mapping Components

i-Seek consists of three principal reasoning components:

a) Commonsense Reasoner: The Commonsense Reasoner uses the above-described semantic networks for doing various types of fail-soft reasoning such as structure mapping, causal likelihood, temporal likelihood, nearest-neighbor situation etc.

b) Planner: The planner coordinates between the front-end Interface, the Reasoner and the Mapping Engine to carry forward the interaction with the user in a seamless manner. The planner maintains the user history, interaction history and mimics the notion of context into derived situations [17]. This way the planner tries to construct a novice user model. For instance, if a Heartburn patient (assuming that *i-Seek* knows about the Heartburn condition) is interacting with *i-Seek* for some advice on weight control and he mentions that he just had tomato-rich food, the Planner can use this information to conjoin with his historical Heartburn condition and can warn about the pitfalls of having acidic food during Heartburn condition.

c) Structure Mapping Engine (SME): The Mapping engine does analogical mapping between expert domain models and the above-described novice models. For instance, from the following simple knowledge pieces,

Novice: Buying lottery tickets is risky

Expert: Investing in stocks is risky,

the SME associates "risk" attribute across "Buying lottery tickets" and "Investing in stocks" concepts.

3.4 The Intelligent Interface

The proposed *i-Seek* system is deployed as a web service and has a simple and intuitive web interface. Users can log into the system and go through series of interactions to get advice about some particular topic within the realms of Online HelpDesk and Personal Finance (Figure 8).

One of the salient features of the interface is providing natural ways of elicitations and explanations. The interface exploits the mapped novice-expert knowledge to elicit knowledge from the user in an indirect and non-obvious way [18,19]. For example, in Figure 8, rather than asking about details of the user's knowhow of computers, ***i-Seek*** abstracts the required knowledge onto some general life activity, in this particular case, traffic. Also, at any stage of interaction the user, if required, can ask the system to explain the rationale behind a particular choice of the system action, which mimics the expert domain model. In order to enable this, the interface provides an adjacent message button next to any feedback or query. For example, when the system tries to gauge the implicit user goal [7] expectation in Personal Finance domain and if the formulated question is unclear to the user, the user can click on the adjacent, "Why are you asking this?" button

and the system provides the following explanation (Figure 9) relating the goal information to the expected return on investments.
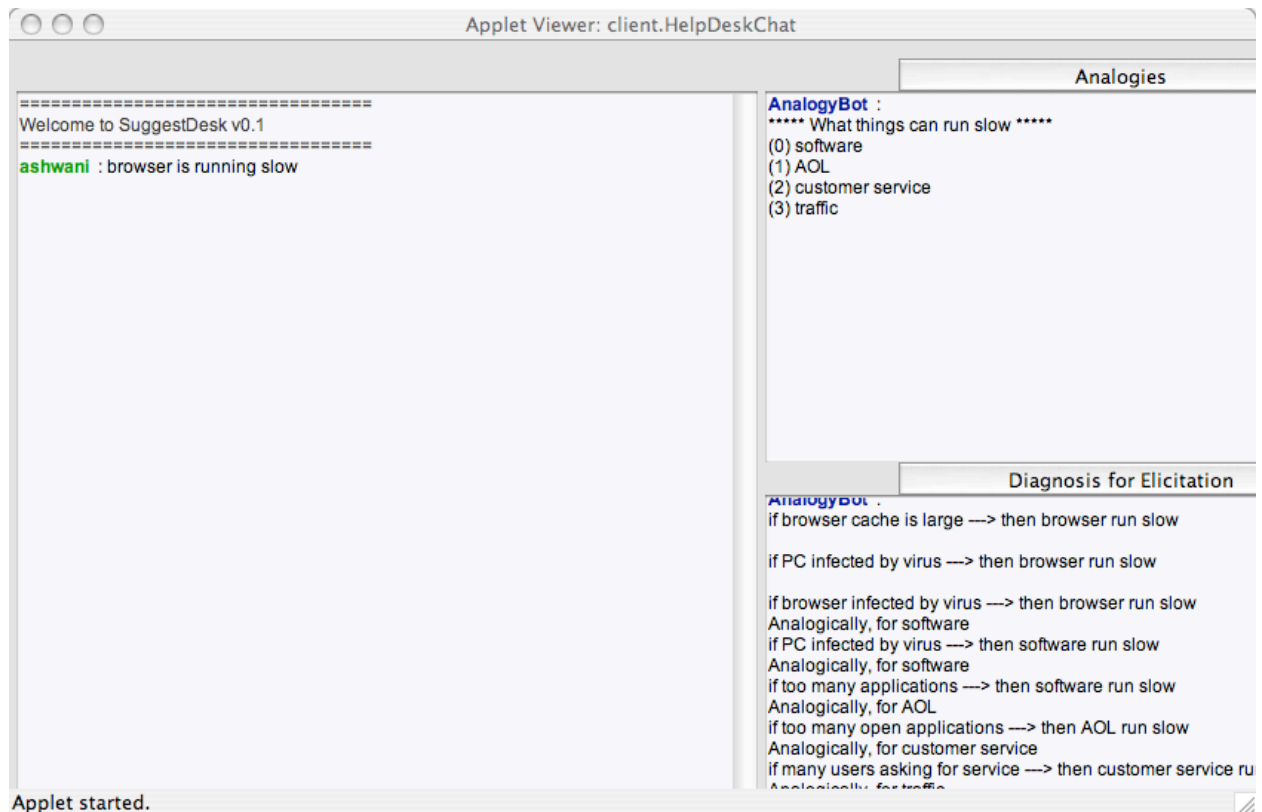
Figure 8: An applet-based web interface for *i-Seek* Online HelpDesk System
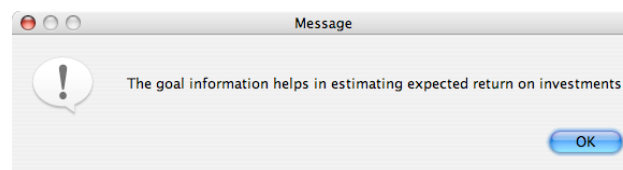
Figure 9: The Explanation Text Box

In the case of Personal Finance domain, some sample domain specific common sense facts used are as follows:

1. 'high risk' -> 'high return'

2. 'high return' -> 'invest in stocks'

3. (PropertyOf "diversified stock" "good growth with high consistency over long term")

4. (PropertyOf "good stock" "larger the growth rate of dividends and earnings")

5 (CapableOf "high stock allocation" "good return for small amount of capital")

Understanding the history states of the interaction and the current goal, *i-Seek* does a sanity check over ensuing user action and reports back to the user if there are any goal-defeating actions from the user. For example, if the user has specified his goal, [I want safe investment] and if he is trying to allocate all his finances to stocks, *i-Seek* alerts the user about potential consequences of such a strategy. In this case, it uses the following common sense knowledge piece:

(CapableOf "high stock allocation" "risky investment"),

and infers that the user goal state, [I want safe investment] conflicts with the potential result state, [risky investment].

4 *i-Seek* Commonsense Paradigm

When people make analogies, they perceive some aspects of the structures of two situations -- the essence of those situations in some sense -- as identical. Essentially, making an analogy requires highlighting various different aspects of a situation and the aspects that are highlighted are often not the most obvious features. Consider two analogies involving DNA [20]-- the first is the analogy between DNA and a zipper. When we are presented with this analogy, the image of DNA that comes to mind is that of two strands of paired nucleotides (which can come apart like a zipper for the purposes of replication.) The second analogy involves comparing DNA to the source code of a computer program. What comes to mind now is the fact that information in the DNA gets compiler into enzymes, which correspond, to the machine code.

As it is apparent from these cases, no single, rigid representation can capture what is going on in a particular analogy-making. In the contexts of different analogical mappings, very different facets of this large representation structure are selected out as being relevant, by the pressures of the particular context.

Essentially, analogical process can be broken into 2 categories:

a) situation-perception, which involves taking the data involved with a given situation and filtering and organizing them in various ways to provide an appropriate representation for a given context..

b) mapping, which involves taking the representations of two situations and finding appropriate correspondences between components of one representation with components of other representation to produce the match-up.

This task becomes very difficult in case of interactive applications that act as experts as source and target knowledge are highly disparate and it is highly probable that there might not be any obvious shared structures between the two knowledge pieces. Hence, while designing mixed-initiative novice-expert systems, full formalization of all potential relevant knowledge may not be cost-effective or practical [21].

Also, mixed-initiative systems present many challenges in terms of user interface designs. It is highly desired to understand the task at hand, make suggestions in context, and present information in a manner, which is understandable to the user [22]. The more the system knows the more helpful it can be and hence (see Table 1), it might seem that the systems should be completely formalized. However, formalizing everything is challenging in its own right and without a guarantee of tractability. Ideally, mixed-initiative systems could formally represent and reason with subsets of the problem that can be formalized, while leaving other parts of the problem to the humans and their more thorough understanding of the task.

Models / Properties	Expert Models	User/Novice Models	Commonsense Models
Depth of Knowledge	Deep and detailed knowledge about the domain specifics.	Shallow knowledge about the specific domain.	Shallow knowledge about the domain and everyday life.
Breadth of Knowledge	Narrow knowledge about the domain.	Narrow knowledge about the domain.	Narrow knowledge about the domain and broad knowledge about everyday life.
Reasoning	Mostly, unidirectional reasoning and inference mechanisms.	Limited or ad-hoc reasoning.	Fail-soft reasoning and analogy mapping.
Usability	Domain knowledge usable only to experts.	Novice knowledge but only usable by experts.	Analogies usable to both experts and novices.

Table 1: Novice, Expert, and Commonsense frameworks

Forming an analogy involves mapping elements from a particular situation onto elements in a separate situation in a way that preserves the relationships between the elements in each situation [4]. Representations that analogies are constructed between are semi-formal structures having predicate-argument structures. called *csfragment*.

(*EffectOf 'download applications' 'browser infected by virus'*)

(*EffectOf 'download plugins' 'hacked by hackers'*)

(*EffectOf 'hacked by hackers' 'browser doesn't start'*)

Analogy is always constructed between 2 *csfragments*: one called *source* while the other *target*. The *source csfragment* is a construct that is already present in the system while the *target csfragment* represents new input to the system. Constructing an analogy between two *csfragments* involves finding a set of mappings between their constituent arguments and associated relationships.

Gentner and Markman [23] distinguish between feature-based alignment as similarity and relation-based alignment as analogy. We do not construe analogy in a strict sense as for mixed-initiative systems both feature-based and relation-based alignments are important. More over, our representation is flexible enough to accommodate both features and relations in a unified framework of semi-formal structures.

In order to find suitable alignment mappings across *csfragments*, each argument from the *source csfragment* must be compared to each argument in the *target csfragment*. This comparison involves attempting to align the source arguments' peer relationships with the target arguments' peer relationships. A naïve approach to alignment would involve matching each of the relationships that the source argument features in with each of the relationships that the target argument features in. Consequently, the process of forming a complete alignment is prone to combinatorial explosions in terms of both arguments (due to exhaustive argument-to-argument comparisons) and inter-argument relationships (as a greater number of relationships will require a greater number of comparisons during the structure alignment.)

We assume that is not necessary to make every possible comparison between the relationships present in the *csfragments* as networks of relationships contain some

redundancy e.g symmetric, transitive relations. Following, this our algorithm aligns pairs of arguments based on the relations common between pairs of arguments in each *csfragment*. Once aligned pairs of arguments have been added to the alignment they are not considered during the rest of the alignment process. This approach tames the complexity problem by ignoring potentially redundant relationships. The graph algorithm restructures the semantic graph by spreading activation and uses thresholding to prune the network of analogous nodes. This, our approach to analogy involves identifying the structural roles played by arguments in the *source* and *target csfragments* and then determining the similarity of these roles. If the roles are similar, the fragments are aligned by creating a mapping between them.

Our approach to analogy allows any alignment between two pairs or related arguments to be ascribed a value representing the strength of alignment. This strength is calculated by the `get_analogies` function mentioned below:

```
get_analogies(concept)
-inputs a concept node
-uses structure-mapping to generate a list of
analogous concepts
-each analogous concept shares some structural features
with the input node
-the strength of an analogy is determined by the number
and weights of each feature. a weighting scheme is used
to disproportionately weight different relation types
and also weights a structural feature by the equation:
log(f+0.5*i+4), where f= outgoing edges
i = incoming edges
- outputs a list of RESULTs rank-ordered by relevance
- each RESULT is a triple of the form:
          ('analogous concept', SHARED_STRUCTURES, SCORE)
- SHARED_STRUCTURES is a list of triples, each of the form:
```

```
              ('RelationType', 'target node', SCORE2)
- SCORE2 is a scalar valuation of the strength of a
particular shared structure
```

The SME also uses `get_context` function from ConceptNet to bring out contextually relevant concepts to the specified argument-predicate tuple.

```
get_context(textnode_list, max_node_visits=500, max_results=200,
flow_pinch=300, linktype_weights_dict=None,
textnode_list_weighted_p=0)
the max_node_visits determines how far context will spread
increasing it adversely affects runtime
max_results limits the number of results returned
but changing it does not affect runtime
flow_pinch limits the number of edges considered
at each step of the context flow
the linktype_weights_dict is a python dictionary
whose keys are the conceptnet relationtypes and whose
values are a weight assigned to each, in the range [0.0,1.0]
- to blacklist a linktype, set its weight to 0.0
- context flow along backedges are regulated by entries in the
linktype_weights_dict whose key names are "relationtype"+"Inverse"
- considering inverse flows slow the runtime of this function a bit
- omitting a relationtype will default to it being blacklisted
- for reference, the default_linktype_weights_dict is:
    default_linktype_weights_dict = {

        'ConceptuallyRelatedTo':0.1,
        'IsA':0.9,
        'FirstSubeventOf':1.0,
        'DesirousEffectOf':1.0,
        'ThematicKLine':0.8,
        'MadeOf':0.7,
```

```
'SubeventOf':0.9,
'UsedFor':1.0,
'SuperThematicKLine':1.0,
'DefinedAs':1.0,
'LastSubeventOf':1.0,
'LocationOf':0.9,
'CapableOfReceivingAction':0.6,
'CapableOf':0.8,
'PrerequisiteEventOf':1.0,
'MotivationOf':1.0,
'PropertyOf':1.0,
'PartOf':1.0,
'EffectOf':1.0,
'DesireOf':1.0,
'ConceptuallyRelatedToInverse':0.0,
'IsAInverse':0.0,
'FirstSubeventOfInverse':0.0,
'DesirousEffectOfInverse':0.0,
'ThematicKLineInverse':0.0,
'MadeOfInverse':0.0,
'SubeventOfInverse':0.0,
'UsedForInverse':0.0,
'SuperThematicKLineInverse':0.0,
'DefinedAsInverse':0.0,
'LastSubeventOfInverse':0.0,
'LocationOfInverse':0.0,
'CapableOfReceivingActionInverse':0.0,
'CapableOfInverse':0.0,
'PrerequisiteEventOfInverse':0.0,
'MotivationOfInverse':0.0,
'PropertyOfInverse':0.0,
'PartOfInverse':0.0,
'EffectOfInverse':0.0,
'DesireOfInverse':0.0,
```

```
        }
if textnode_list_weighted_p, then each element of textnode_list
is not a string, but instead, of the form: ('dog',0.5)
where the cdr is the relative origin weight of that concept.
```

Finally, the matching algorithm uses the `project_details` function to bring out hierarchical structures, if any within a particular argument-predicate tuple.

```
project_details(textnode_list)
-inputs a list of concepts
-computes the detail projection, which consists of
a thing's parts, materials, properties, and instances
and an event's subevents
-returns a rank-ordered list of concepts and their scores
e.g.: (('concept1',score1), ('concept2,score2), ...)
```

4.1 Personal Finance

There exist numerous investment tools [24,25,26,27,28,29,30] that claim to come up with the best strategies for asset allocation through a sequence of questions to gauge the user's inclination towards investment and willingness to take risks. But in our research what we found was the lack of sufficient control to the user, limited personalization and limited usability scenarios. The tools need a richer interactive experience. These tools do not seem to exploit commonsense knowledge to either achieve the user's goal or to make the interaction more natural. Common sense, as we understand it in today's world is shared knowledge that puts everyone on the same page and provides an enormous closeness to human thought, hence making communication easier and more intuitive. Common sense can be used to explain the success and failure of different scenarios and help troubleshoot problems along the lines of Woodstein [31], an interactive debugger. It is similar to Woodstein in providing system feedback at all stages of processing, however it differs from Woodstein in the respect that Woodstein didn't make any analogical reasoning to provide these elicitations and explanations.

Some of the primary motivations for this system are as follows:

a) Goals and Motivations play a big part in financial decisions as "mental accounting" research in behavioral economics emphasizes [32,33];

b) Poor saving habits and skewed risk attitudes [34,35];

c) Existing Investment tools:
   - Lack explanations/rationales, interactivity, customizability;
   - Limited usability scenario, and
   - Lack Common Sense.

In this project there is an attempt to bridge the gap between naïve and expert knowledge systems by providing an intuitive interactive framework where the user can interact with the system using natural language sentences without being overwhelmed by the expert knowledge processing that the system performs. For instance, lay users cannot objectively specify their risk tolerance, as they may not be aware of the repercussions of taking low or high risk, so it is essential for the tool to engage in a dialogue and gauge the users' risk tolerance.

Besides, the system uses a goal-oriented Concrete Budgeting framework, which has been developed with Professor Dan Ariely. The central premise is that people are irrational spenders and if a tool can provide means to account for different real-life goals of a person, it can help the person in making appropriate saving plans. Users can specify different type of investment accounts, which are differentiated using different timeline requirements to achieve the account goals [see Figure 10 below]. These accounts from Concrete Budgeting are mapped to the associated concepts in ConceptNet. For instance, an account like "Auto" is mapped to the contextually relevant concepts of "car", "vehicle", "road" etc in ConceptNet for further processing.
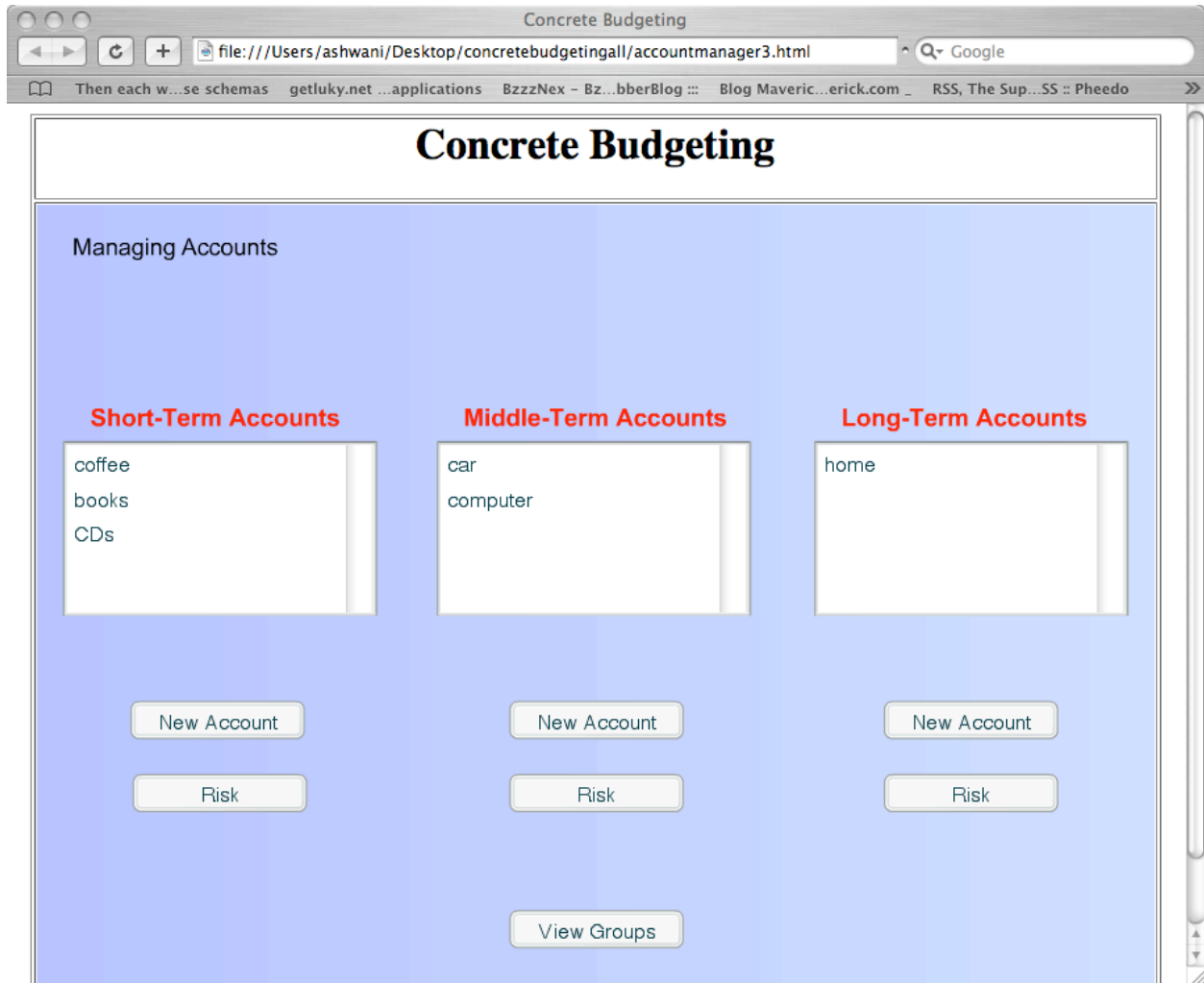
Figure 10: Concrete budgeting Interface for adding and editing accounts

In the following section, we describe the functional architecture followed by brief descriptions of various underlying components in subsequent sections.

4.1.1 InvestAssistant Architecture and Components

The system architecture involves the following components:

• Common Sense Analyzer (CSA)

- Expert financial engine

- InfoFilter -Information Filter

- OMCS Interface

- Investment Strategies
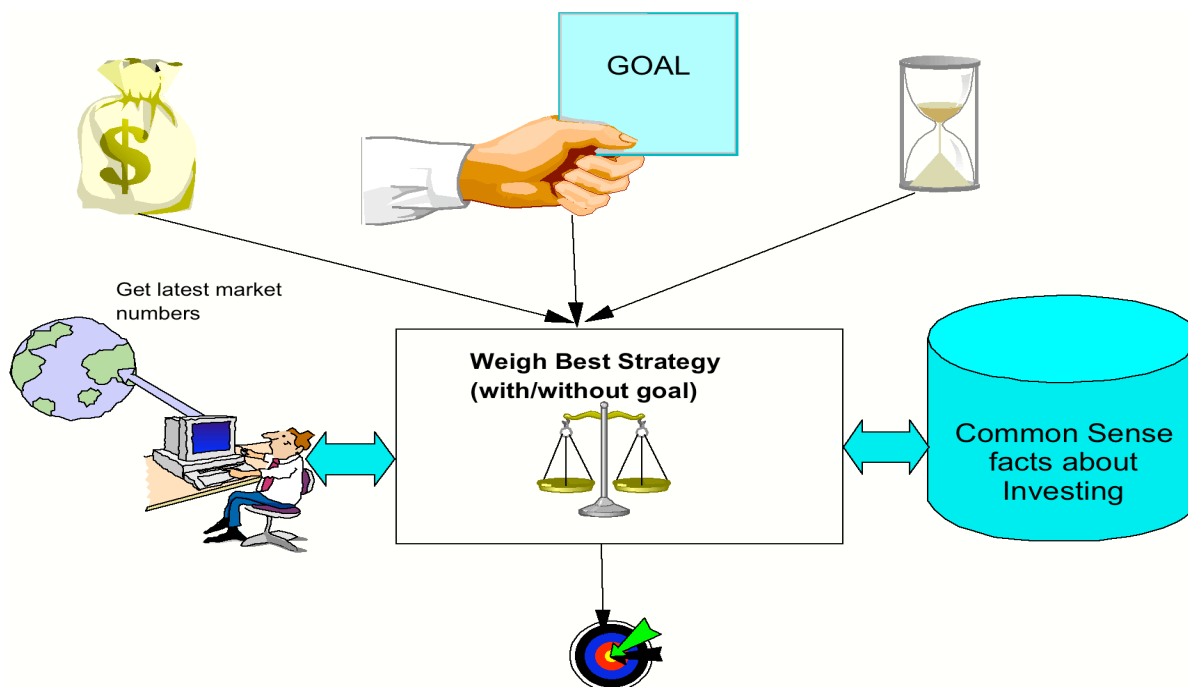
Figure 11: InvestAssistant functional architecture

The investment strategist interprets the analyzed request from the CSA and accumulates relevant information from the InfoFilter and Expert Financial Engine. The CSA plays a crucial role in bridging the gap between natural man-machine interactions and expert system processing. CSA comprises of a Natural Language Understanding front-end, which processes the user's commands in natural language to extract investment and goal

semantics. This abstract level semantics is correlated with the common sense knowledge base in order to establish various goal and action dependencies. The CSA implements an interface, which interacts with the OMCSNET using SOAP (Simple Object Access Protocol) messages. OMCS is configured to run as a web service, which is queried to extract semantic associations in the form of predicates.

4.1.2 SYSTEM DESCRIPTION

The input to the system contains the investment amount, timeline and intended purpose. The output is suggestions for asset allocations and best individual asset picks. The system contains a natural language dialogue framework, an in-built browser and a user-interface to exchange information with the system and to retrieve explanations and history of interaction, all interfaced to a common-sense database.

4.1.2.1 Common Sense Analyzer

Contemporary research in the area of interactive goal-driven systems has emphasized the importance of having a dialogue-based interaction as opposed to fixed menu or scenario based interactions with the user [10]. However, having dialogues in the mode of natural languages requires that the system have adequate language understanding capabilities, fail-soft inference and deduction mechanisms. It is imperative that the system has sufficient common sense knowledge and optimal application-specific knowledge i.e. expert knowledge. From the usability point of view, it is also desired to maintain a seamless and intuitive interface that bridges these two different types of knowledge pieces.

In InvestAssistant, our central goal is to achieve this kind of interactivity, without sacrificing application performance or overloading the end user with the application specific modus operandi. The key idea is to specify suitable mappings from natural language utterances to expert system behaviors and vice versa. The important thing to note here is that these mappings are dynamic in the sense that they evolve with

interactions, they are personalized based on the user's profile, and they get refined as the common sense knowledge base gets richer.
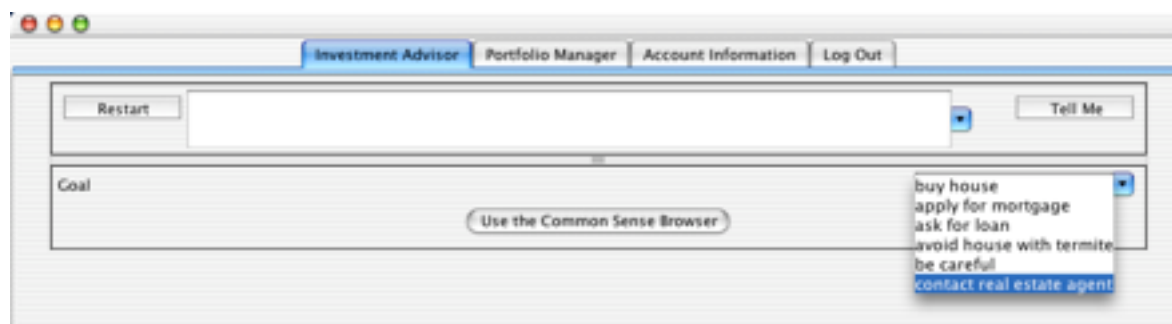
Figure 12: Commonsense information about goal
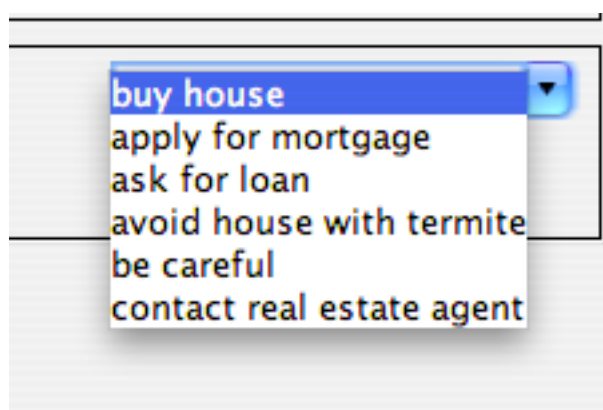
Figure 13: Related concepts to "buy house"

4.1.2.2 Natural Language Understanding (NLU) Unit

All the user's requests are first tagged using a Parts of Speech (POS) Tagger. The tagged text is chunked using a text-chunker, which groups tagged words within an utterance to disjoint classes based on some pre-defined rules. Further, a semantic analyzer produces the semantic parse of the sentence in the form of an n-ary argument structure. (Refer to Figure 14)

The semantic parse obtained in this manner specifies the actual action semantics for the application. One of the key derivations is the frame structure that is built upon this semantic parse. Based on the verbs occurring in the semantic parse and respective synonyms, the NLU unit constructs a frame-based semantic structure [36,37,38,39], which is then correlated with the lexical predicates in the ConceptNet (see Figure 16).

-------------- Tagging User Request -----------

I/PRP want/VBP to/TO invest/VB 1000/CD dollars/NNS

-------------- Chunking User Request -----------

(NX I/PRP NX) (VX want/VBP to/TO invest/VB VX) (NX 1000/CD dollars/NNS NX)

-------------- Semantic Parse of the request in the form: (Verb-Subj-Obj-Obj) -----------

("invest" "I" "1000 dollar")
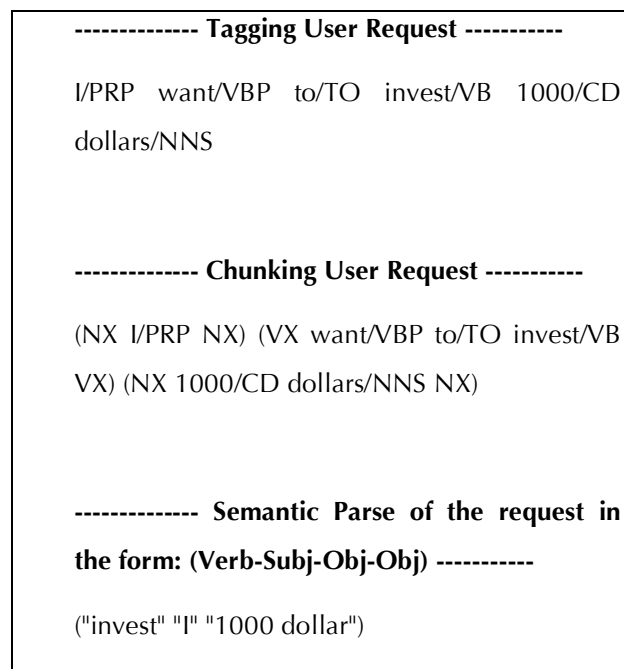
Figure 14: Semantic Parsing

The frame structure comprises of hierarchical event-object structures derived from the semantic parse and chunked-text. This kind of generic type-based construction has subsequent positive implications on goal planning and iterative interaction with the ConceptNet [40].

4.1.2.3 Action Planning
The system needs to map the derived semantics from the user's utterance to the intentional goal structures and in turn to its own application level goal planning. As the investment-

strategy process is iterative and state-based, it seemed appropriate to model it using finite state automata, where states are characterized by the various steps needed to lay out an investment strategy and the transitions encode various choices that the user can express using natural language.
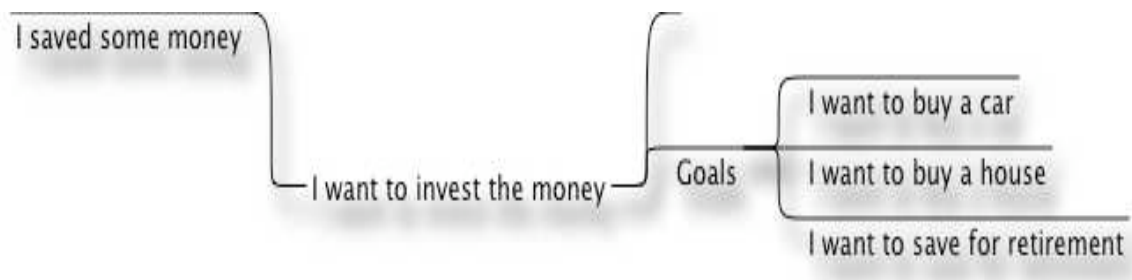
Figure 15: Dependent Actions map

Essentially, goals have slot-filler type structures and by progressing through the state automata, it is made feasible to attain the level where adequate investment advice could be extracted from the expert system. For instance, refer to Figure 16 for the frame structure for the "invest_goal".

```
<<<Frame Name: invest>>>
        Type : event
        Subject : I
        Objects:
        Object 1 :<<obj1>>
                Type : dollar
                Attributes :
                Attribute 1:<<attr1>>
                     Attribute_Name:
                individuation
                         Attribute_Value     :
1000
```
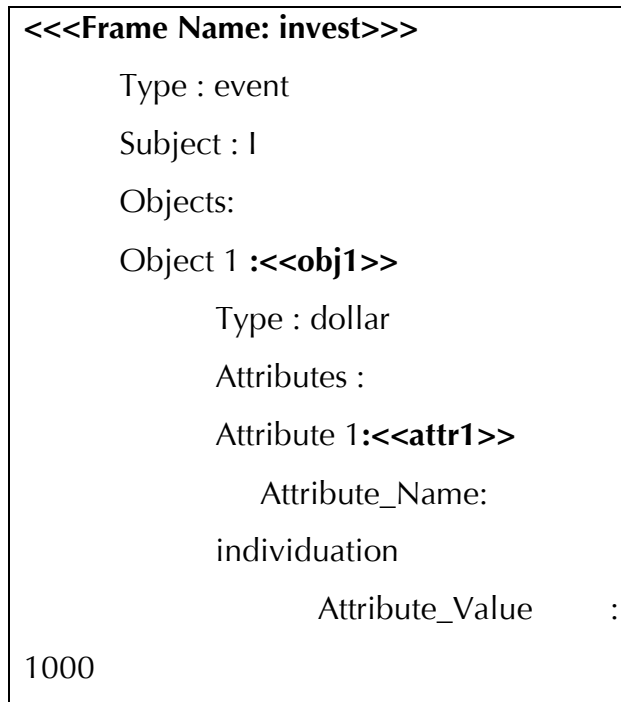
Figure 16: Frame Representation for "invest money"

```
<<<Frame Name: buy>>>
        Type : event
        Subject : USER
        Objects:
        Object 1 :<<obj1>>
                Type : THING
                Attributes :
                Attribute 1:<<attr1>>
                Attribute_Name: TEMPORAL
                Attribute_Value : _time_value
```
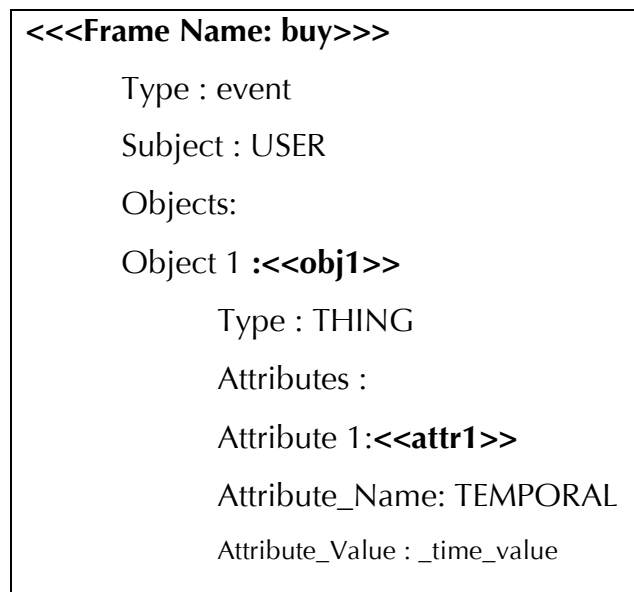
Figure 17: Frame-based slot Representation for "buy"

Similarly, the invest_action requires an "invest" frame, where the slots pertaining to the investment object is filled with the money to be invested. Naturally, maintaining frame semantics of an utterance has advantages as the utterance frames can be compositionally correlated with the action frames (for example, subsumption criteria), thus providing a computationally efficient and an incremental approach to collaborative goal-oriented action planning and goal execution.

4.1.2.4 Common Sense Inference

One of the key issues that we address is that of decomposition of semantics into simpler structures that are efficient from a computational standpoint. Frame semantics is an elegant framework for characterizing fully specified semantics. However, due to the inherent ambiguity and potential for multiple senses, it becomes essential to correlate the fully specified frame semantics to the relevant senses. Also, from the action planning point of view it is necessary to articulate necessary and sufficient steps to achieve the desired goal. The common sense knowledge fulfills both of these requirements as it encodes multiple senses in a semantic network, where traversal along a particular path could reflect various steps needed to complete a particular goal [16].

As mentioned earlier, we use OpenMind [1] as the source for the common sense knowledge. OpenMind is a web-based collaborative project that aims towards acquiring knowledge in the form of English sentences that we use in our day-to-day activities. This knowledge is structured in a semantic network that specifies predicate-based semantics.

Therefore, we construct relevant queries pertaining to the user's goal and accounts [Figure 18], which are used to gather other senses of the goal as well as other goals, which are required to achieve the goal. For instance, a typical OMCS query 'buy house', produces binary predicate structures:

- (EventForGoalEvent "buy house" "apply for mortgage")

- (EventForGoalEvent "buy house" "ask for loan")

- (EventForGoalEvent "buy house" "avoid house with termite")

- (EventForGoalEvent "buy house" "be careful")

- (EventForGoalEvent "buy house" "contact real estate agent")

- (EventForGoalEvent "buy house" "contact your local real estate agent")

Figure 18: Goal-oriented investment account information
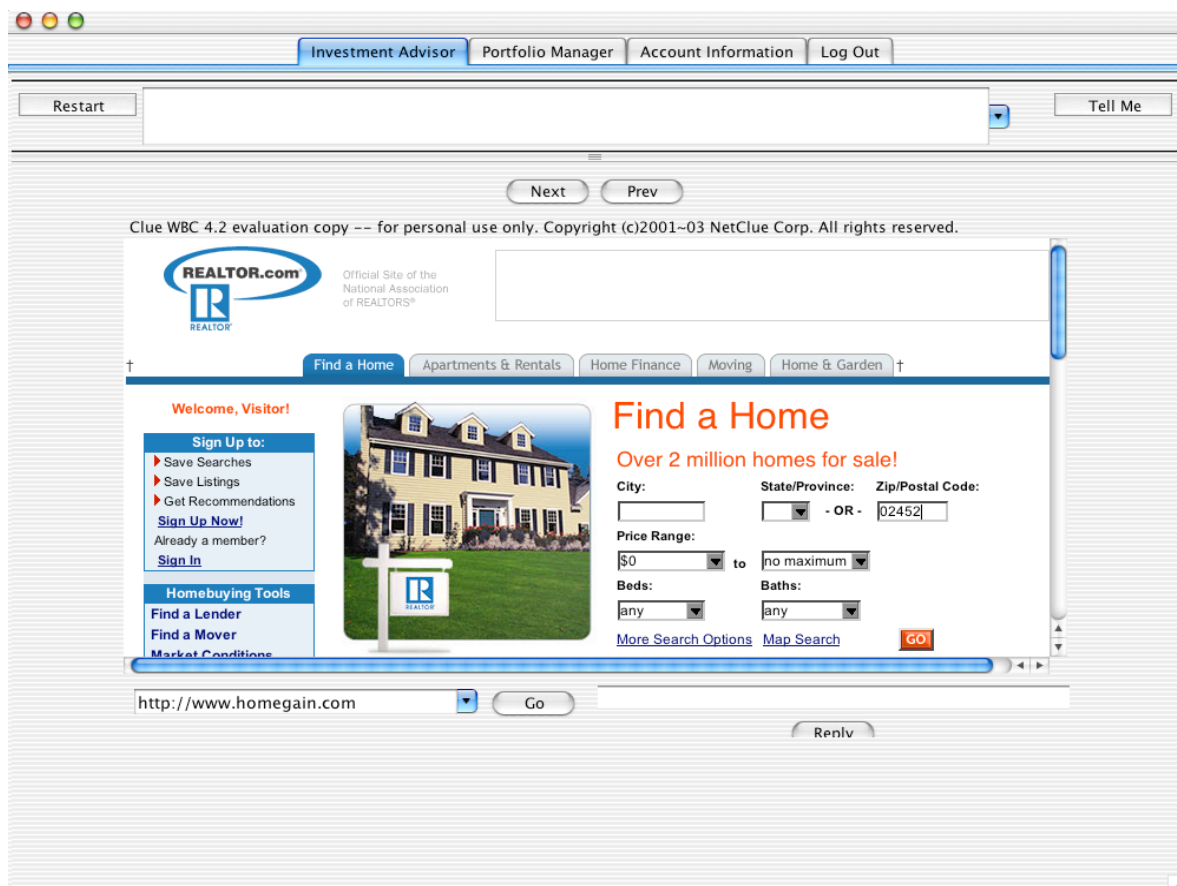
Figure 19: Google Interface to expert sites

4.1.3 Moving Into the Expert Domain

At this point we do some handholding with the user to better define the goal. We split this phase into three parts.

First, we use the concepts as queries and crawl the web to get the most relevant links that offer information about the goal. So, from the earlier example of "buy house", common sense comes back with facts like "real estate". The links that InvestAssistant returns will pertain to contacting real-estate agents and buying a house. The web provides a wealth of well-conducted research on various topics hence offering the expertise required to narrow down the goal. We carefully extract the best links and display it in a menu along with an in-built browser for the user to navigate.

Second, the user now navigates the web to get more information about the goal. While this is happening, our tool is "listening" to the hyperlinks. When the user finally closes on a price or value it is passed to the system and in the backend the current URL is captured for two reasons. One is to be able to return to the site at a later point either for debugging purposes or to redo the selection. Two is to extract other options that the tool can suggest to the user, if the current choice was not good.

Part three of our expert system is an information filter where we have an agent that goes out to the web looking for financial information particularly pertaining to making investments. The search is intelligent in that it looks at different industries and companies and extracts the factors that affect the performance of the market, like the volatility, price-earnings ratio, etc. At the point this filter is triggered, the agent has at its disposal the asset-allocation determined earlier. So the input to the filter will be the expected performance of the various investments (like stocks, bonds, money market) in order to meet the user's goal and if necessary make a profit. Subsequently, the system provides mechanisms to specify expected timeline of investment and intuitive way to set and edit risk levels associated with all the user investment accounts [Figure 20].
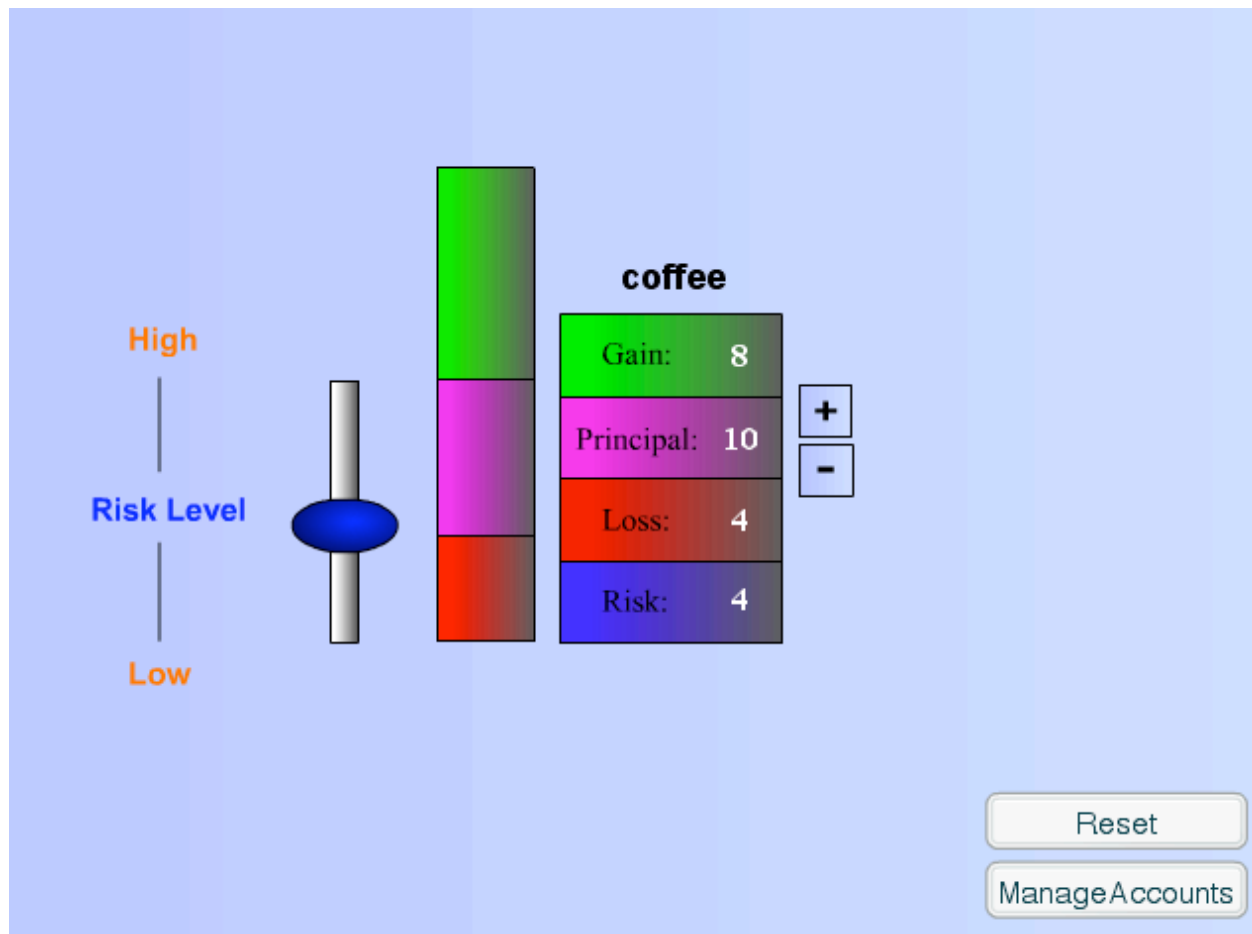
Figure 20: Risk profiling for individual investment accounts

4.1.4 Common Sense Investment Strategy

The initial asset allocation is determined from the users' goal, timeline and risk tolerance. Now, we delve into each allocation and use a combination of commonsense [7] and expert knowledge to pick the top performing industries and companies the user may consider investing in and we explain the reasons behind making this selection. The typical domain-specific expert common sense facts used are as follows:

- 'high risk' -> 'possible high return'

- 'high return' -> 'better chance by investing in stocks'

- (PropertyOf "diversified stock" "good growth with high consistency over long term")

- (PropertyOf "good stock" "larger the growth rate of dividends and earnings")

- (CapableOf "high stock allocation" "good return for small amount of capital")
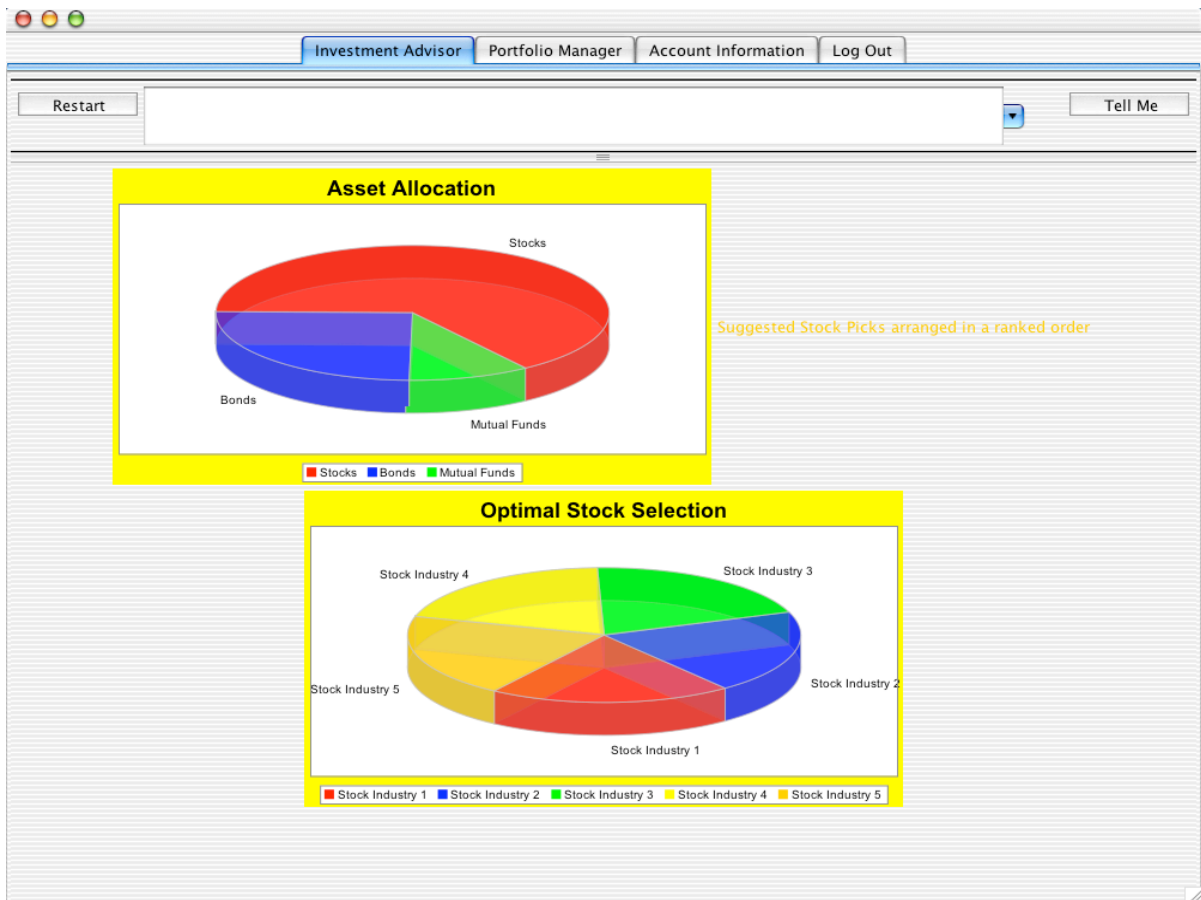
Figure 21: Asset allocation chart
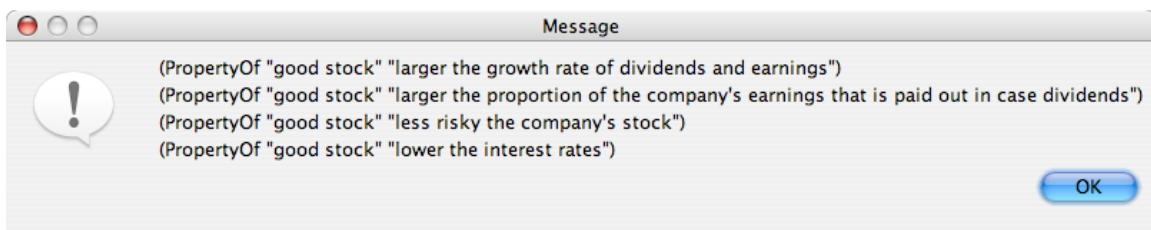
Figure 22: Expert Commonsense knowledge

4.2 Online HelpDesk

Most of the computer systems have some sort of expert help system for troubleshooting any problems or issues with the provided services. Usually, a help system is a separate module embedded in the system that is oriented to give a quick reference or a task-specific help. Traditionally, these systems are designed to solve problems concerning how to use the system, or to improve user's performance while using it. A sample help advice can be like as follows:

"If you are having trouble connecting to AOL at home, follow the following steps to help guide. You do not have to be connected to AOL for this.

1. Be sure that you have already installed the AOL client onto your laptop.
2. Click on the AOL icon to open the client.
3. Choose the —Help" tab."

Typically, help systems employ manuals to provide advices about domain-specific topics to the user. An online help manual usually has an explorer-like interface and is divided into two panes: the Navigation pane and the Content pane (as can be seen in the AOL HelpDesk, Figure 23 and 24). Some advanced help systems like HEAT [41] go beyond simple FAQ-like formats and provide contextual help pages that address issues with the specific topic. Even though, these systems have good coverage of the domain, they are invariably hard to use by an end user and are unwieldy in the advising process. Most often than not, users do not have much domain knowledge and exposure and hence, cannot understand help descriptions in the domain-specific jargon. Generally, understanding and learning is based on what people do while solving problems. For solving a problem one might need more information or just make use of what one knows already. However, in it is common in existing help systems that after solving a problem how it was solved is lost for others that, very likely, will need to solve the same or a similar problem in the future.

Even more frustrating is the situation in which for a previous problem the use solved, user cannot remember how (s) he arrived at the solution. The prevalent expert systems do not provide any transparent and clear way of sharing their helping process with the user. Moreover, as these systems have limited or no notion of the general user model, their interaction scenarios and explanations are severely limited. It makes difficult for the user to comprehend the expert advisory solution, as there is no way for him to co-relate the domain-specific solution to his day-to-day activities and knowledge.

Additionally, most of the existing help systems have some kind of keyword-based search to locate help topics and associated knowledge material. However, this is also a severe limitation, as domain knowledge cannot be uniquely interpreted by context-free associations of keywords. As can be seen in the Figure 25, when the user asks the AOL HelpDesk system about "how to remove an icon from toolbar", the system returns results about emails rather than icons and toolbar due to inaccuracies in searching and domain-knowledge representation. Indeed, dealing with domain knowledge is a matter of understanding the specific context, domain-related concepts and rules that can be applied to these concepts.

Figure 23: AOL HelpDesk Navigation Pane
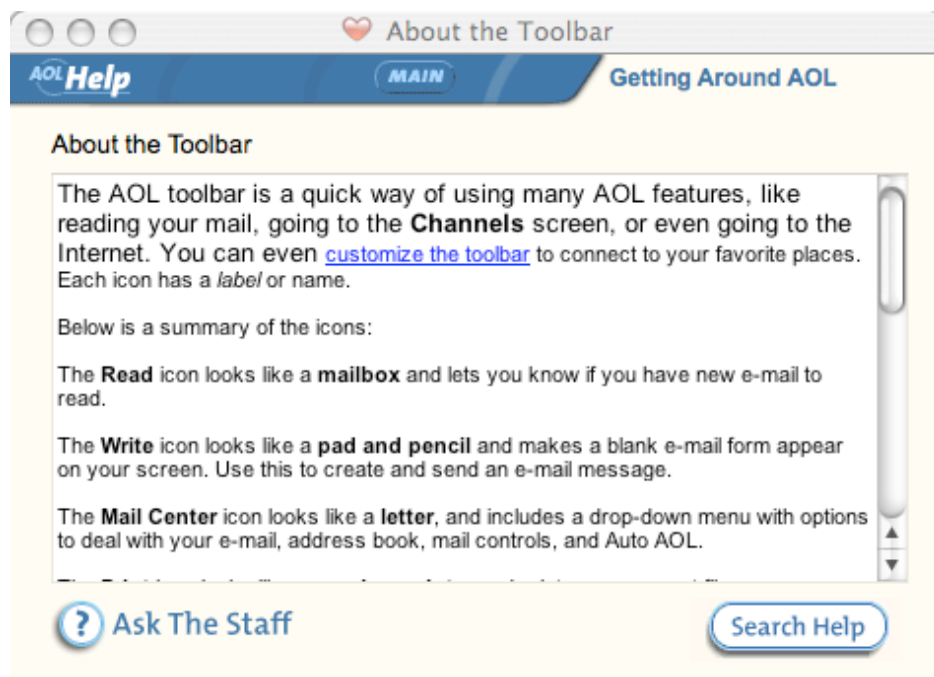
Figure 24: AOL HelpDesk Content Pane

Essentially, the existing help systems do not provide sufficient control to the user, and have limited user models and usability scenarios. These tools need a richer interactive experience in eliciting and explaining knowledge to the user. This would require having some reasonable coverage of user knowledge and providing ways to map the expert knowledge to novice knowledge and vice versa. Commonsense knowledge mined from the OpenMind project provides a rich and substantially wide coverage of novice knowledge, which can be used to bridge the gap between the user and the help assistant.
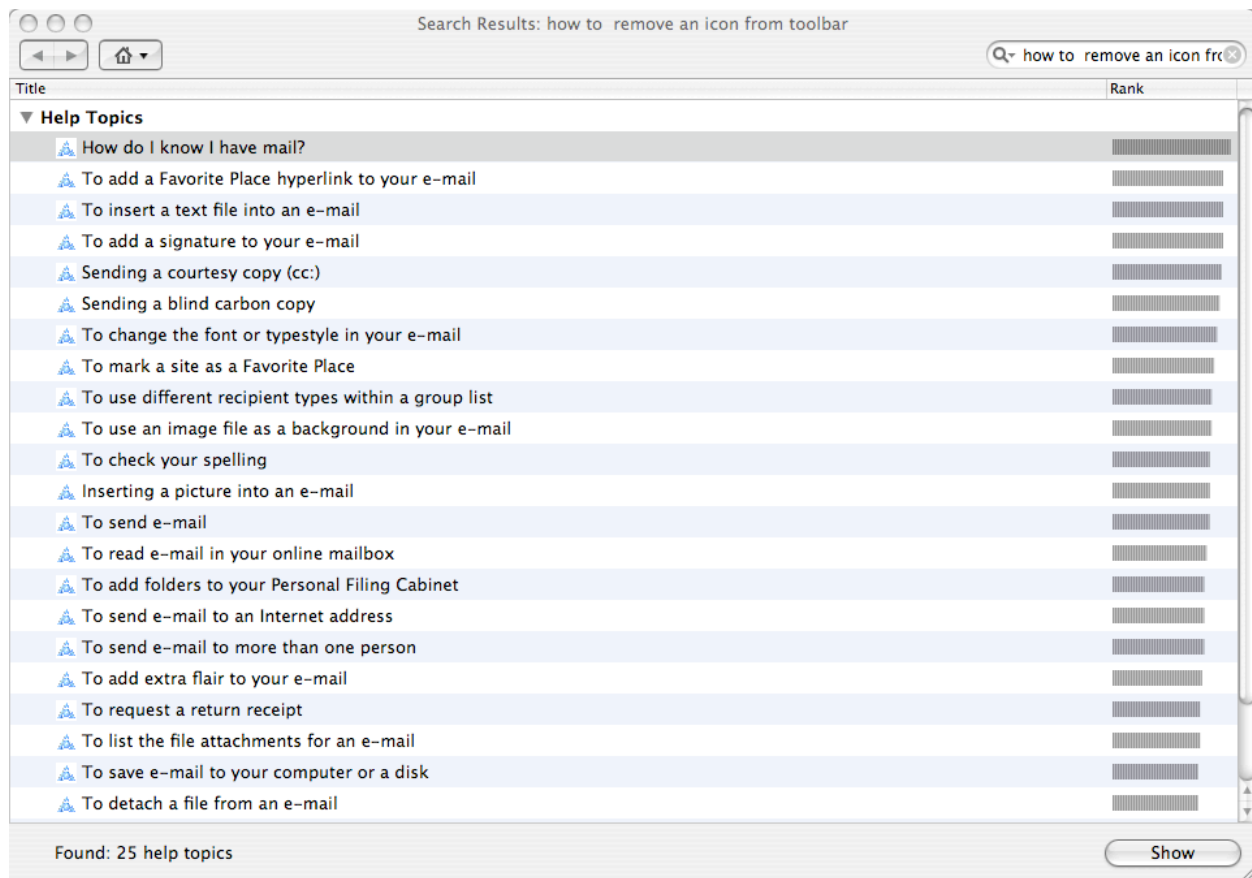
Figure 25: AOL keyword-based search

In this collaborative project with AOL, we have built an interactive chat application called **SuggestDesk**, which attempts to bridge the gap between novice AOL users, who do not have much knowledge about computers and AOL's help knowledgebase by providing an intuitive interactive framework where the user can interact with the system using natural language sentences without being overwhelmed by the expert knowledge processing that the system performs. For instance, the user can chat with the help assistant and communicate that,"browser is running slow", without trying to figure our keywords that would return appropriate results. The system understands the user's utterance and employs structure mapping to deduce relevant analogies, which are provided to the help assistant so that he can make relevant elicitations and explanations about the problem at hand.

In the following are some of the reasons why common sense knowledge can be useful for HelpDesk applications:

    a) Bridge between novice and expert

    b) Plausible elicitations and explanations for specific problems

    c) Scope for dynamic and interactive scenarios by using natural languages

    d) Richer personalization

In the following sub-sections we provide detailed description of the SuggestDesk system.

4.2.1 SuggestDesk: System Description

The SuggestDesk system comprises of the following components:

4.2.1.1 Natural Language Understanding (NLU) module

User can input his problems or issues in natural English and the system employs the NLU module to process the user utterance. NLU uses a shallow language parser, which is able to handle simple English sentences. The parser can understand semantic units in the

utterance. However, the end goal is not to achieve perfect understanding of the meaning of free form text, as this is practically unachievable given the inaccuracies in parsing.

User's inputs are first tagged using a Parts of Speech (POS) Tagger. The tagged text is chunked using a text-chunker, which groups tagged words within an utterance to disjoint classes based on some pre-defined rules. Further, a semantic analyzer produces the semantic parse of the sentence in the form of an n-ary argument structure (see Figure 26 below).

```
-------------- Tagging User Request -----------

browser/NN  is/VBZ running/VBG  slow/JJ

-------------- Chunking User Request -----------

(NX browser/NN NX) (VX is/VBZ running/VBG
VX) slow/JJ

-------------- Semantic Parse of the request in
the form: (Verb-Subj-Obj-Obj) -----------

("run" "browser" "slow")
```

Figure 26: Semantic Parse of "browser is running slow"

The semantic parser also produces additional extracted phrase structures as follows:

```
result = [{prep_phrases_tagged=[], verb_phrases_tagged=[is/VBZ
running/VBG], verb_arg_structures_concise=[("run" "browser" "slow")],
noun_phrases=[browser], noun_phrases_tagged=[browser/NN],
adj_phrases_tagged=[slow/JJ], verb_arg_structures=[[is/VBZ
running/VBG, browser/NN, [slow/JJ]]], modifiers_tagged=[slow/JJ],
prep_phrases=[], verb_phrases=[is running],
parameterized_predicates=[[[run, [past_tense, passive_voice]],
[browser, []], [slow, []]]], modifiers=[slow], adj_phrases=[slow]}]
```

The semantic parse obtained in this manner provides useful semantic chunks in form of the above structures. One of the key derivations is the frame structure that is built upon this semantic parse. Based on the verbs occurring in the semantic parse and respective

synonyms, the NLU unit constructs a frame-based semantic structure [36,37,38,39], which is then correlated with the lexical predicates in ConceptNet and ExpertNet (see Figure 27 below).

```
<<<Frame Name: run >>>
        Type : event
        Subject : browser
        Modifier: slow
        Objects: <>
```
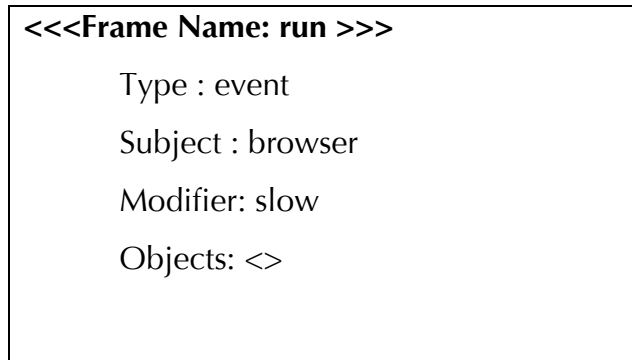
Figure 27: Frame representation of "browser run slow"

The frame structure comprises of hierarchical event-object structures derived from the semantic parse and chunked-text. This kind of generic type-based construction has subsequent positive implications on goal planning and iterative interaction with the ConceptNet [15].

4.2.1.2 SuggestDesk User Interface

The SuggestDesk user interface implements an interactive chat-based client, which both the user and the Help Assistant use. The interface (see Figure 28 on the next page) enables natural language dialogue between the user and the assistant by means of text dialogue boxes at the bottom of the interface. The leftmost pane is used as the primary message window, where both user's and assistant's messages can be seen. This primary pane maintains the complete sequence of user-assistant interaction, until the user closes the client window. On the right hand side are two panes that are only visible to the Help Assistant. This is because analogically mapped knowledge is produced in these windows and if this is exposed to the user (s)he might be overwhelmed by the domain-specific knowledge and might lead to more confusion. On the other hand, the assistant being the domain expert knows precisely how to use this information in order to provide relevant elicitation questions and explanations. The top right pane is used to provide a list of similar objects as the frame structure derived from the user's input based on object attributes and modifier matches. The middle right pane is used to provide analogy-based diagnosis of the problem formulated by the user in context of the objects provided in the top right pane. Thus, the assistant can see similar objects and analogically related diagnosis for the problem at hand and provide the user with better informed answer. Also, the assistant uses the analogies to explain the solution correlating it with some day-today like situation faced by the user. Thus, the interface provides an intuitive and easy way to facilitate natural and seamless dialogue between the user and the assistant.
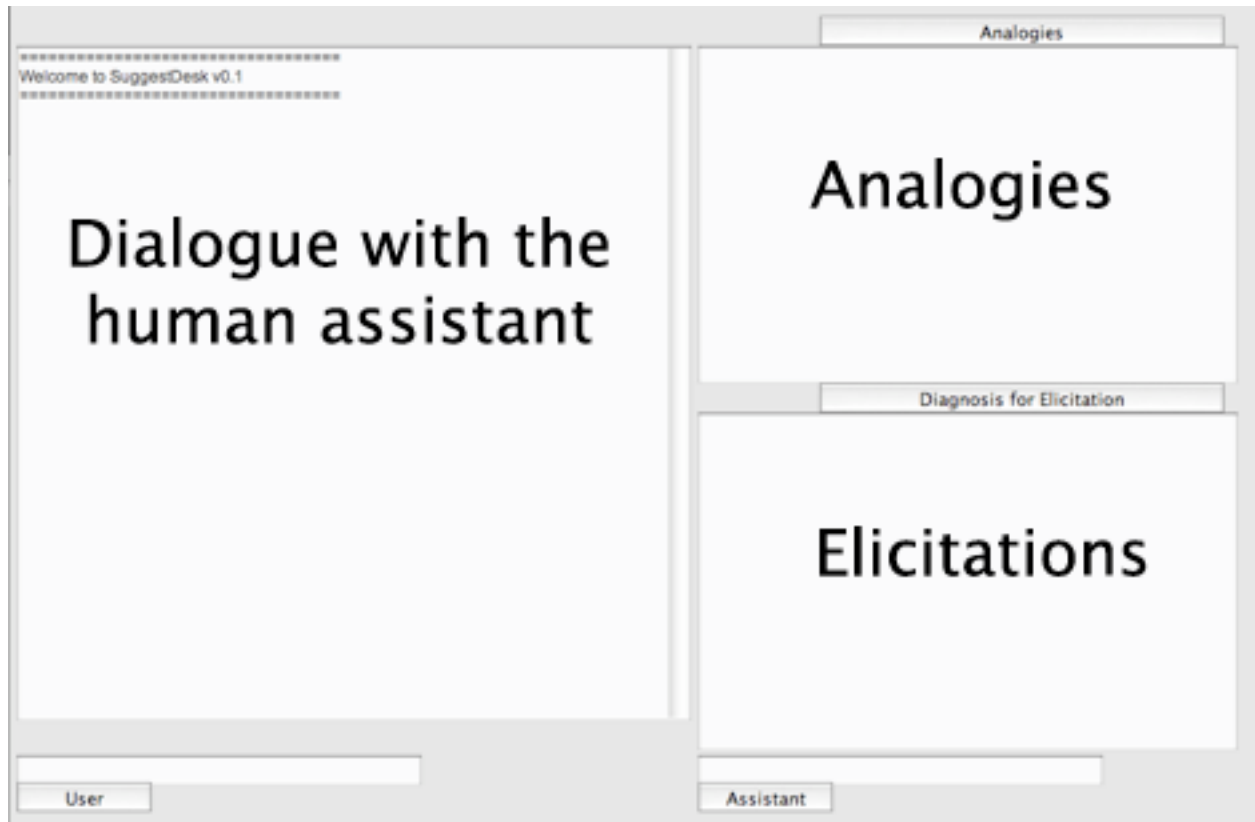
Figure 28: SuggestDesk User Interface

4.2.1.3 Commonsense Processor (CP)

The Commonsense Processor (CP) processes the frame object derived from user's utterance to match it to the novice model constructed using the commonsense knowledge base. Some sample commonsense knowledge about computers looks like as follows:

Author	Knowledge
sai	Computer software usually has lots of bugs
sai	A bug in computer software is an error made by a programmer
eisele	Something you find in software is a programming error (often called a bug)
llarson	Sometimes designing software causes designing software bugs.
wateriswet	Something that might happen while designing software is bugs

Figure 29: Sample OpenMind Knowledge

The aggregate commonsense knowledge in form of these English sentences is processed using the NLU parser and converted into predicate-argument structures. These structures are organized into a semantic graph, where nodes represent the concepts and edges represent relation amongst the concepts. In the following is a sample of such structures, where f is the number of outgoing edges, while i is the number of incoming edges:

```
(IsA "family" "group of person" "f=2;i=0;")

(CapableOfReceivingAction "story" "contain" "f=0;i=10;")

(LocationOf "water" "in toilet" "f=2;i=0;")

(UsedFor "write" "communication" "f=2;i=0;")

(LocationOf "trash" "in dumpster" "f=4;i=0;")

(CapableOfReceivingAction "exposure" "extend" "f=0;i=4;")

(UsedFor "work of art" "admire" "f=2;i=0;")

(UsedFor "computer" "compute" "f=2;i=1;")
```

4.2.1.4 Expert Analyzer (EA)

The Expert Analyzer (EA) builds a semantic network of domain knowledge in a similar way CP builds the novice semantic network. EA uses the AOL Help knowledge base (see Figure 30 on the next page) to mine help topics related to key concepts in the help domain, such as the following:

> *Browsers download pictures and files to computer.*
>
> *It helps in faster reload of the webpages.*
>
> *These files are known as browser cache or Temporary Internet Files.*
>
> *After sometime, the cache size may build up and cause the browser to slow down if it is not cleaned.*
>
>
> *Browsers can be vulnerable to viruses. some free applications can have viruses.*
>
> *Viruses use browser's resources. This may cause the browser to run slowly.*

EA employs the NLU unit to parse and chunk these topical sentence fragments into predicate-argument type semantic units. These structures are organized into a semantic graph called ExpertNet, where nodes represent domain-specific concepts and edges represent the relations. For instance, the ExpertNet for help domain has the following structures related to Internet and browsers:

```
(IsA 'internet explorer' 'browser')

(CapableOf 'browser' 'download files')

(CapableOf 'browser' 'download applications')

(EffectOf 'surf internet' 'download files')

(EffectOf 'surf internet' 'download applications')

(EffectOf 'download files' 'browser cache is large')

(EffectOf 'download applications' 'browser infected by virus')

(EffectOf 'hacked by hackers' 'browser doesn't start')

(EffectOf 'PC infected by virus' 'browser run slow')
```
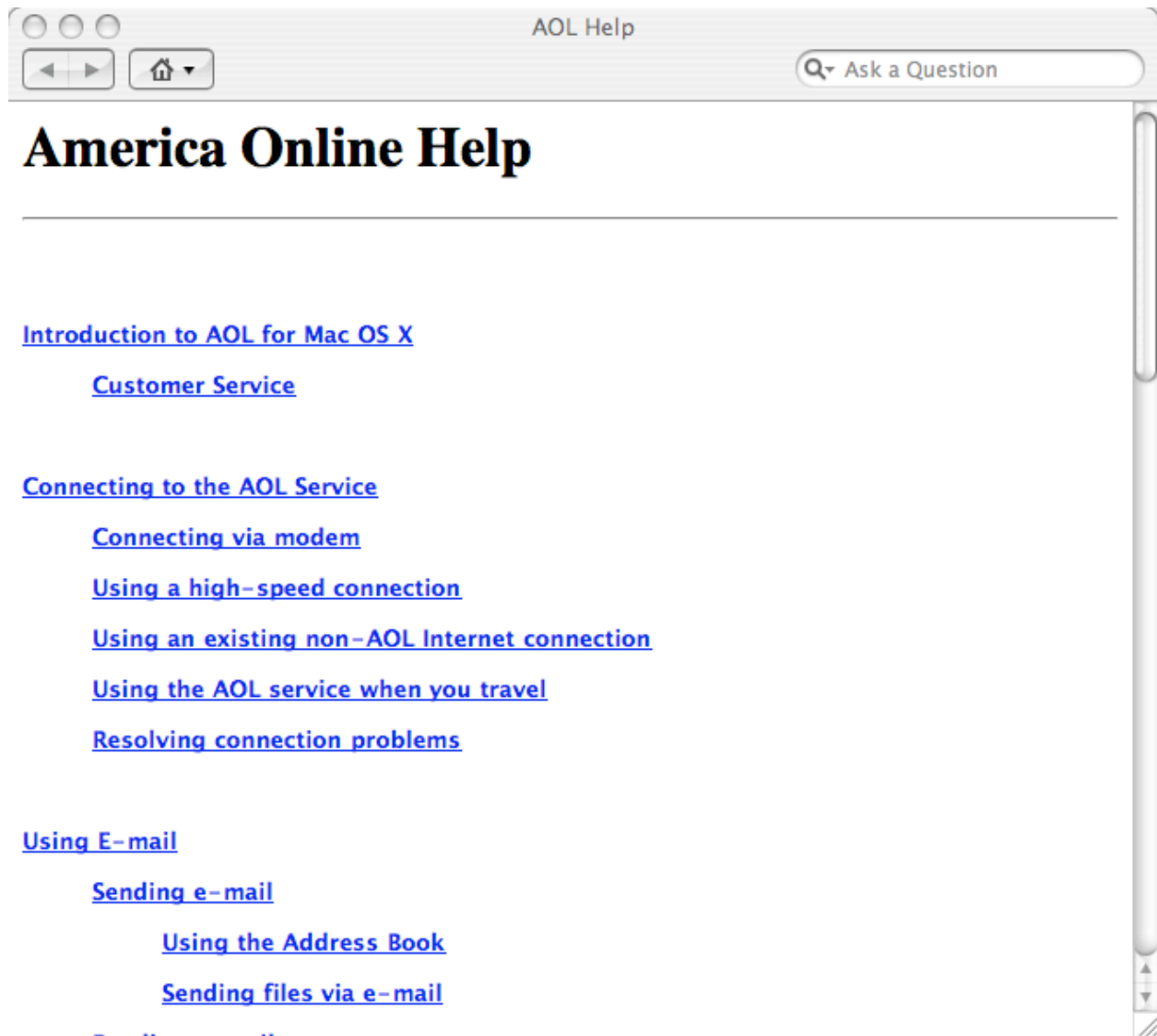
Figure 30: AOL's knowledge base interface

4.2.1.5 Analogy Mapping Engine (AME)

The Analogy Mapping Engine (AME) uses the ConceptNet and ExpertNet as constructed above to perform novice-expert knowledge mapping. Since, both the ConceptNet and ExpertNet are similar in graphical structure, the AME is able to perform fast and efficient graph matching algorithm. AME implements a variation of the Structure Mapping Algorithm to align the two graphs and matches concepts in both the networks depending upon node attributes and respective relations. Subsequently, AME looks at the precise frame description of the user problem to perform matching in a hierarchical manner. For instance, in the example of, [[browser], [run slow]], AME first aligns both graphs using the verb, [run] and further, computes the similarity based on modifier relations, like in the following sample result:

```
Analogies:[[computer, [[UsedFor, surf internet, 1.1887218755408673],
[CapableOfReceivingAction, run slow, 1.1887218755408673],
[CapableOfReceivingAction, crash, 1.1887218755408673],
[CapableOfReceivingAction, start, 1.1887218755408673]],
6.1887218755408675], [car, [[CapableOfReceivingAction, damage,
1.188721875408673], [CapableOfReceivingAction, crash,
1.188721875408673], [CapableOfReceivingAction, start,
1.188721875408673]], 5.930167946706389], [software,
[[CapableOfReceivingAction, run slow, 1.1887218755408673],
[CapableOfReceivingAction, crash, 1.1887218755408673],
[CapableOfReceivingAction, install, 1.1887218755408673],
[CapableOfReceivingAction, install, 1.1887218755408673]],
5.855516191543203]]
```

AME provides a ranking mechanism for the analogous structures as specified by the following function:

```
get_analogies(concept)
-the strength of an analogy is determined by the number
and weights of each feature. a weighting scheme is used
to disproportionately weight different relation types
and also weights a structural feature by the equation:
log(f+0.5*i+4), where f= outgoing edges
i = incoming edges
```

4.2.1.6 Elicitation and Explanation Processor (EEP)

The Elicitation and Explanation Processor (EEP) retrieves the analogies from AME and processes the ranked list of analogies that match the given user problem. It analyzes each analogy and looks at the structure relations in order to construct diagnostic elicitations. For instance, for the "browser is running slow" example, EEP retrieves the list of possibly analogous objects matching browser for the "running slow" property (see Figure 31 below).
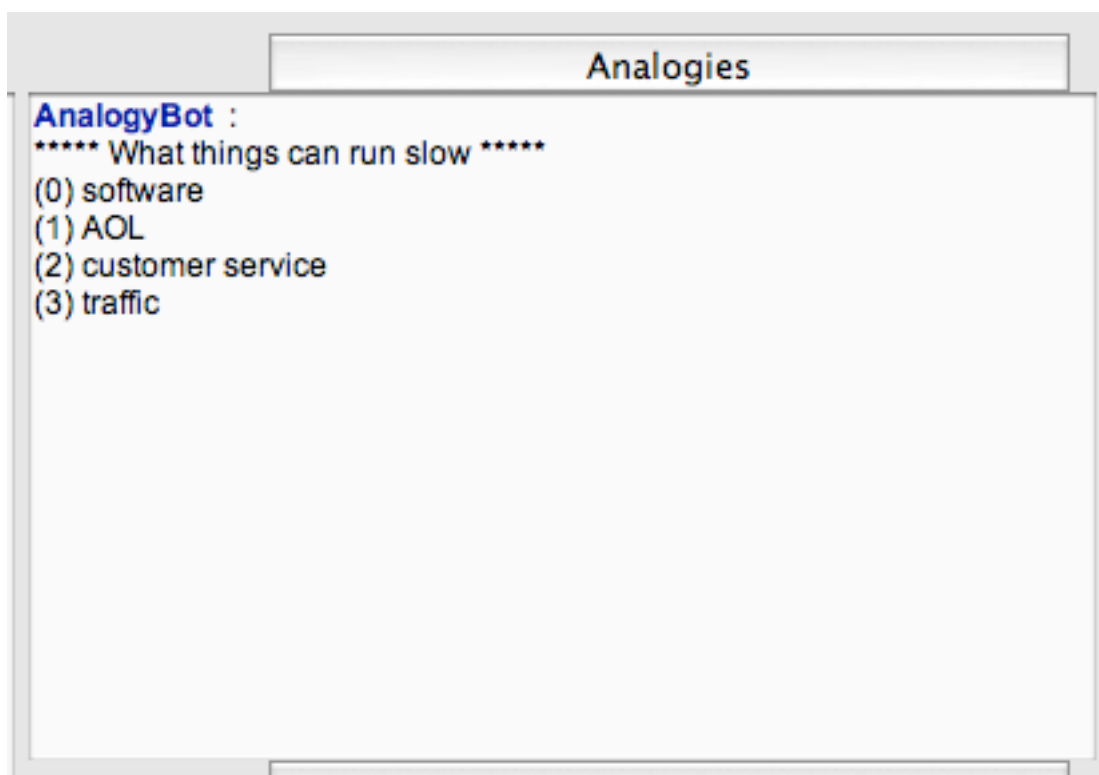
Figure 31: Analogy Interface

EEP processes every concept in this list to enumerate causes for this property using the ConceptNet and ExpertNet. After retrieving the likely analogous causes, it outputs the analogies and associated diagnoses in the "Diagnosis for Elicitation" window (see Figure 32 on the next page). Thus, EEP enables delivery of analogies and associated diagnostic information to the SuggestDesk UI.

The Figure 33 illustrates a complete interaction scenario for the "browser is running slow" scenario, where EEP provides relevant analogies, which is used by the assistant to elicit and explain domain-specific knowledge in novice's terms.

Diagnosis for Elicitation

AnalogyBot :
if browser cache is large ---> then browser run slow

if PC infected by virus ---> then browser run slow

if browser infected by virus ---> then browser run slow
Analogically, for software
if PC infected by virus ---> then software run slow
Analogically, for software
if too many applications ---> then software run slow
Analogically, for AOL
if too many open applications ---> then AOL run slow
Analogically, for customer service
if many users asking for service ---> then customer service run slow
Analogically, for traffic
if many vehicles on the road ---> then traffic run slow

Figure 32: Analogy-based Diagnoses for similar concepts
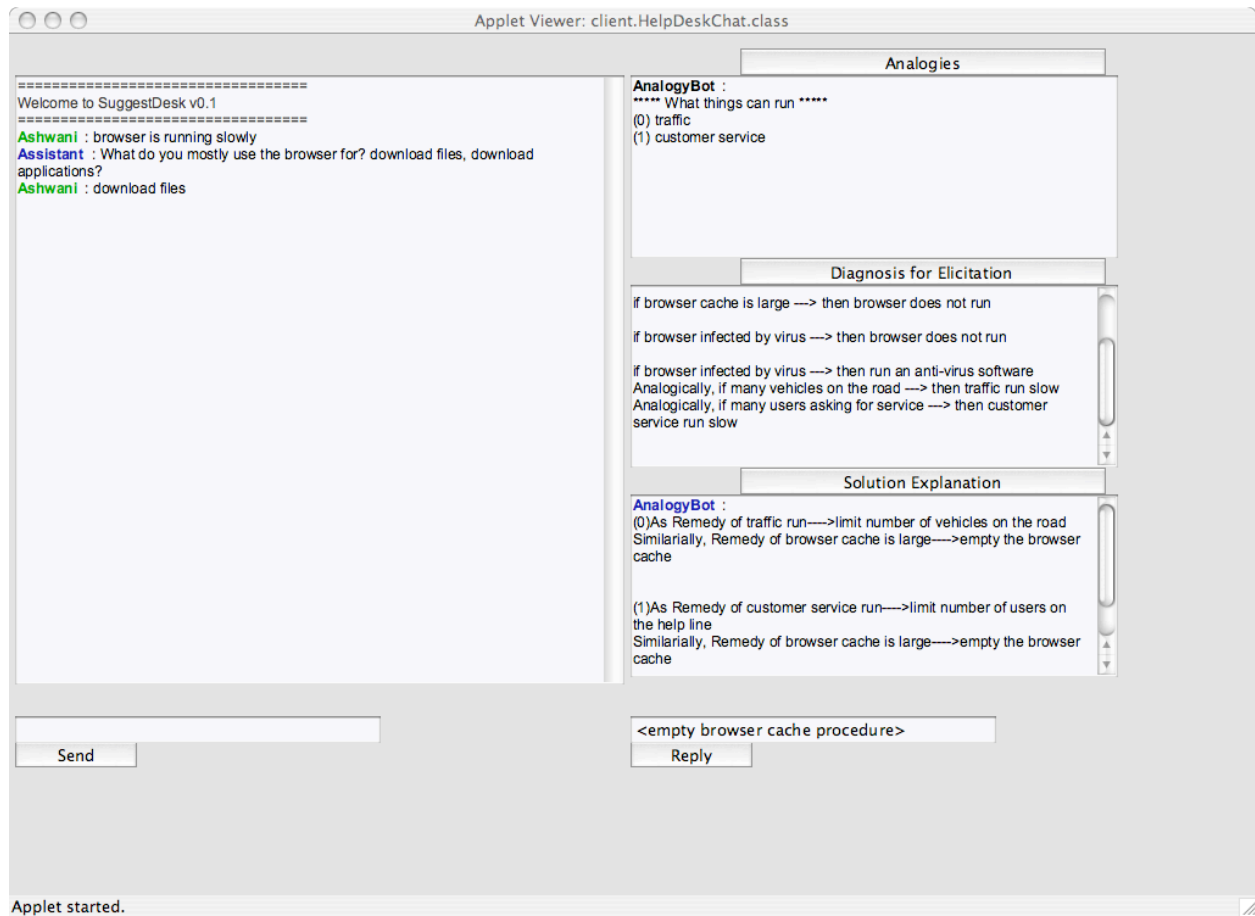
Figure 33: SuggestDesk complete interaction

5 Implementation

The i-Seek system is implemented in Python and Java. Additionally, it uses XML-RPC to communicate to the ConceptNet and ExpertNet servers.

Some of the principal components and snippets of implementation are illustrated in the following:

1) XML-RPC Server

Steps:
a) Importing ConceptNet database

```
import ConceptNetDB
...
```
b) Importing Predicates file
```
pred_filename = "predicates.txt"
```

c) Loading ConceptNet predicates
```
print "Loading Predicates from %s..."%pred_filename
c =ConceptNetDB.ConceptNetDB(None)
..
xmlrpc.serve_forever()
```

2) Analogy Mapping Engine

Steps:
a) Definition
```
def
get_analogous_concepts(self,textnode,simple_results_p=0):
     decode_node,encode_node,encode_word,decode_word =
self.decode_node,self.encode_node,self.encode_word,self.dec
ode_word
```

b) Encoding the text node
```
textnode=textnode.strip()
encoded_node = encode_node(textnode)
```

c) Searching and Structure Mapping algorithm implementation
```
for fe in fes:
    commonpred,commonnode,f,i = fe
    if commonpred in linktype_stoplist:
        continue
    bes = bw_edges.get(commonnode,[])
    bes = map(edgeuid2zipped,bes)
```

```
                   bes = filter(lambda x:x[0]==commonpred and
       x[1]!=node_uid,bes)
             for be in bes:
                   commonpred2,candidate,f2,i2 = be
                       link_strength =
       math.log(f+f2+0.5*(i+i2)+2,4)
                   weight =
       link_strength*linktype_weights.get(decode_word(commonp
       red),1.0)
```

3) get_context from Java

Steps:
a) Function Definition

```
       def get_contextFromJava(self,
       textnode,max_node_visits=500,max_results=200,flow_pinch=300
       ,linktype_weights_dict=None,textnode_list_weighted_p=0):
             textnode=textnode.strip()
             textnode_list = string.split(textnode," ")
             return
       self.get_context(textnode_list,max_node_visits,max_results,
       flow_pinch,linktype_weights_dict,textnode_list_weighted_p)
```

4) Data Structure for Expert knowledge piece

```
public class Expert
{
     String relation="";
     String arg1="";
     String arg2="";
     public static void main(String[] args) {
     }
}
```

5) Processing Analogies

Steps:
a) Function Definiton

```
private LinkedList processAnalogies(String analogies, String subeve)
```

b) Extracting Analogies

```
                   if(analog.charAt(i) == ',')
                       {
                           count++;
```

```
                            if(count == 2)
                                index1 = i;
                         }
                    if(analog.charAt(i) == '[')

                       {
                        count1++;
                        if(count1 == 3)
                        {
                            index2=i;
                                                anentity =
                 analog.substring(index2+1, index1);

                            aclist.add(anentity);
                            break;
                      …
```

6) Processing User's Request

Steps:
  a) Function Definition

```
          public void processRequest(String un, String message)
```

  b) Setting up XML-RPC server connection
```
       String server_url = "http://localhost:8000";

                    // Create an object to represent our server.
                    SimpleXmlRpcClient  server = new
    SimpleXmlRpcClient(server_url);


                       // Build our parameter list.
                    Vector params = new Vector();

              // Call the server, and get our result.

                    params.addElement(message);
```

c) Calling the ConceptNet XML-RPC server and invoking NLU unit
```
       try {
            p.addElement(server.execute("nltools.generate_extraction",
            params));
               } catch (IOException e2) {
                    // TODO Auto-generated catch block
                    e2.printStackTrace();
               }
```

```
            String subj =
(server.execute("nltools.jist_entities", p)).toString();

              subj = (String) subj.subSequence(1,
subj.length()-1);

        String  eve= "";

if(((List)(server.execute("nltools.jist_subj_events",
p))).size() >0)
        eve = (server.execute("nltools.jist_subj_events",
p)).toString();
```

d) Processing Expert domain

```
        for(int i=0;i<aclist.size();i++)
            {
             in++;
             if(in >20) break;
             String analogsubj =  (String)aclist.get(i);
             analog += "\n"+ "("+i+") "  + analogsubj;
             for(int j=0; j<exlist.size();j++)
                 {


                  ao.subj = analogsubj;
                  ao.prob = analogsubj+ " " + subeve;

if(((eveindex1 > -1) || (eveindex2 > -1)) &&
(subeve.length() > 0))
                          {

if(rel.equalsIgnoreCase("EffectOf"))
                              diag += "Analogically, for
"+ analogsubj+ " \n if " + arg1 + " ---> then   " +
arg2+"\n";
                          ao.EffectOf = arg1;
                          analoglist.add(ao);
```

6 Evaluation

The primary goal of the *i-Seek* system is to provide a clear and intuitive interface for non-expert users to seek domain-specific information. To evaluate the efficacy of the system, the collaborator at AOL has conducted empirical user evaluation of the *i-Seek* system. Besides performing experiments at MIT, we thought it would be useful to do evaluations in an industry setting as well. The central premise to be evaluated is whether advisory systems advice with commonsense-enabled elicitation and explanation framework are better than those without the commonsense knowledge in terms of user experience, task completion, and usability.

Hypothesis

The hypothesis to be tested was that the users of *i-Seek* in presence of the commonsense model would execute tasks in shorter times and with better precision than with omission of the commonsense model and as compared to other available expert systems. We also hypothesized that by using the *i-Seek* interface users would be better informed about a domain-related topic. Finally, we contend that the commonsense elicitation and explanation model would receive higher satisfaction ratings than the existing expert system interfaces.

Purpose

The purpose of the experiments was to compare *i-Seek* with the existing AOL HelpDesk system. We wished to determine that which system and associated interface was best for different tasks, what interface features the users preferred, and how the commonsense framework affects the system behavior. The independent variable was the type of the system used (such as *i-Seek,* AOL Help etc) while the dependent variable was the task execution time, success rates, and subjective satisfaction. To measure the task execution variable, the subjects received tasks sequentially and bookmarked the concluding advisory

page. If the user was not convinced with the system's advice, (s)he carried on the interaction. At any stage, the user could mark the task as complete or failed and proceed to the next task. Thus, errors were factored into the study as higher task execution time. To measure the success rate variable, the user after concluding the task marked successful or failed against the task entry on a paper. For the subjective satisfaction variable, users filled out a user satisfaction questionnaire upon completion of the experiment.

Design

The evaluation user group consisted of 1 AOL collaborator and 4 MIT students. We collected both qualitative and quantitative data as part of the user surveys after every interaction with the system. This included measures such as task completion times, success rates and overall satisfaction with and without the *elicitation-explanation* model. There were 2 task scenarios, one related to browser performance issue, and the other one related to computer crashing problem. To start with the users filled out a preliminary questionnaire. Subsequently, the users were provided with an introduction and a training routine for both i-Seek and AOL HelpDesk system. The details of the training routine varied from interface to interface, but each session consisted of a demonstration of all features of the interface as well as dummy task scenarios. The subjects could ask any questions at any time during the training routines. After the training, users were asked to perform 2 practice task, similar in nature to the experimental tasks – one with i-Seek, and another with the AOL HelpDesk system.

The breakdown of the experiment was as follows:
* Preliminary Questionnaire - 5 minutes
* Training - 10 minutes
* Tasks - 20 minutes
* User Satisfaction Survey - 10 minutes

Results

Task 1	Avg.Completion Time (in minutes)	Success	Avg. Satisfaction (from 1 - 10)
AOL HelpDesk	10	Success (3/5)	5
i-Seek SuggestDesk	4	Success (5/5)	9.0

Table 2 : Results for Task 1, browser performance issue

Task 2	Completion Time	Success	Satisfaction (from 1 - 10)
AOL HelpDesk	12	Success (2/5)	2
i-Seek SuggestDesk	5	Success (4/5)	7.5

Table 3: Results for Task 2, computer crashing issue

As the results indicate in the above two tables, on average i-Seek's SuggestDesk fared better than AOL's HelpDesk in terms of average task completion time, success rate, and average satisfaction score. Moreover, 4 out of 5 users liked the analogies provided by the system.

7 Related Work

Our research within the i-Seek project touches various aspects of contemporary research in Artificial Intelligence. Very broadly, our work can be related in some aspects to the following three sub-domains of AI:

7.1 Knowledge Acquisition

Knowledge acquisition has been a challenging area of research in AI, with its roots in early work to develop expert systems. In the early systems, knowledge acquisition was predominantly a manual process in which a human would encode domain knowledge in form of rules. However, as the systems got more and more complex, new techniques and interfaces were developed so that a domain expert could input the knowledge, which was further encoded into rules automatically. Although there has been considerable work in this area, activities have been distributed across several distinct research communities. For instance, in machine learning, learning apprentices acquire knowledge by non-intrusively watching a user perform a task, while in planning, mixed-initiative systems acquire knowledge about a user's goals by taking commands or accepting advice regarding a task.

Several notable works in the knowledge acquisition domain such as EXPECT [42] and TRELLIS [43] fall into the latter category. EXPECT is a rich system that uses ontologies and knowledge acquisition scripts to generate and advance dialogues with users to acquire and maintain knowledge bases of a diverse nature. Our approach is similar to EXPECT in the aspect that we use the acquired novice knowledge from the user to map it to expert domain and produce suitable *elicitations,* which reflects the system's understanding of the user's context. TRELLIS is an interactive web-based application for argumentation and decision-making. The systems supports the user to create knowledge "snippets" from online information resources. The key is to capture how the user progressively generates new knowledge that results in added value to the original raw information sources. As

EXPECT and TRELLIS systems realize that each user can have unique requirements of a system, they provide means for modifying task representations using constraint-satisfaction. Depending on the specific context of the user interaction, certain constraints are considered to be predominant. These systems use these constraints to elicit more detailed information from the user. This way these systems enable a meta-reasoning framework, which can deal with composition of knowledge fragment in order to achieve a task, account for missing information, and context. Elicitations are in form of text questions that hide the inner system syntax and interfaces provide Browser-and-replace functionality for task. However, both EXPECT and TRELLIS systems are limited to elicitations only and provide very little information, if any of the rationale behind the specific task procedure undertaken by the system. Essentially, it is over-simplistically assumed that the user and the system share knowledge structures, which in many real life situations don't hold true, especially in the context of expert advisory systems. Our research has shown that it is highly useful when the system is able to mediate the mapping from novice to expert so that the mapping primitives and structures could be reused not only for *elicitation* but also for *explanation* purposes.

Some of the other knowledge acquisition issues related to our research have also been dealt by Timothy Chklovski in the LEARNER system, where the system learns by example knowledge pieces entered by users. Our expert and novice OMCS mimic similar behavior in terms of how users input the knowledge. However, we go beyond simple aggregation of knowledge by organizing the knowledge pieces in a semantic graph structure, which helps in performing efficient reasoning methods.

7.2 Intelligent Tutoring Systems

Traditional Computer Aided Instruction systems used to be inefficient, and expensive. Recognizing its deficiencies, Intelligent Tutoring systems were developed which attempted to adapt the speed and level of interactivity to that required by a student. Early Intelligent Tutoring Systems implemented variations of rule-based expert systems. These early

systems:

- Contained a component with expertise in teaching
- Contained expertise in the task being taught
- Maintained a model of what the student has understood or possibly misunderstood

Although, there is no standard architecture for an ITS, four components emerge from literature as part of an ITS [44,45,46,47]. These are *the expert model*, *the student diagnosis model*, and *the curriculum model, and the instruction model.*

Much like a domain expert, the expert model in an ITS has in-depth knowledge about a particular domain. Traditionally, this knowledge is both factual and procedural and is maintained by an expert system. Factual knowledge represents information about the problem domain, while procedural knowledge contains knowledge of task procedures and rules that an expert uses to solve problems within that domain. The facility in the ITS for sequencing and selecting problems is the curriculum manager. To select the appropriate problems for the student, the curriculum manager extracts performance measurements from the profile stored in the student model. Like a human instructor, an ITS coaches the student through the use of an Instructional Environment, which facilitates *explanations*. The instructional environment provides the student with tools for proceeding through a tutorial session and obtaining help when needed. It also determines when the student needs unsolicited advice and triggers its display.

i-Seek also maintains expert and novice models along the lines of Intelligent Tutoring systems, but goes beyond simple diagnostic *explanations* that ITS systems are capable of providing. As ITS systems have narrow coverage of user model or the domain is severely limited, these systems are not able to *elicit* domain-independent feedback from the user. However, in i-Seek we have broader commonsense knowledge and narrow domain knowledge available, which enables it to perform fail-soft reasoning. In case the system doesn't find any relevant domain information, it falls back on the general commonsense knowledge and *elicits* useful information from the user. We believe that i-Seek architecture

could be ideal for an Intelligent Tutoring system as well.

7.3   Analogy-based Expert Systems

Most of the interactive applications that try to map disparate knowledge pieces employ some form of analogical reasoning. Indeed, Analogy-making is crucial for human cognition. Many cognitive processes involve analogy-making in one way or another: perceiving a stone as a human face, solving a problem in a way similar to another problem previously solved, arguing in court for a case based on its common structure with another case, understanding metaphors, communication emotions, learning, translating poetry from one language to another [48]. All these cases require a suitable mapping to be established between two cases or domains based on their shared structures and common systems of relations.

Analogy-making is a very basic cognitive ability, which appears to be present in humans from a very early stage and develops over time. It starts with the simple ability of babies to imitate adults and to recognize when adults are imitating them, progresses to children's being able to recognize an analogy between a picture and the corresponding real object, and ultimately, culminates in the adult ability to make complex analogies between various situations. This seems to suggest that analogy-making serves as the basis for numerous other kinds of human thinking and explains the important of developing computation models of analogy-making.

Most of the existing work in this area can be divided along the following process of analogy-making:

Representation

> This process is absent in most models of analogy-making. Typically, hand-made representations are fed into the model. However, there are some models (ANALOGY, CopyCat, TableTop, MetaCat) that do produce their own high-level

representations based on essentially unprocessed input. These latter models [49,50,51] attempt to build flexible, context-sensitive representations during the course of the mapping phase. Other models, such as AMBR [52] perform re-representation of old episodes

Retrieval

There have been extensive studies experimentally and it is now clear that superficial similarity plays the major role in analogical retrieval i.e. the retrieval of a source for analogy is easier if it shares similar objects, similar properties, similar general theme with the target. Structural similarity, the familiarity of the domain from which the analogy is drawn, the richness of its representations and the presence of generalized schemas also facilitate retrieval. Most models of retrieval are based on exhaustive search of LTM and on the assumption that old episodes have context-independent encapsulated representations. There are, however, exceptions (e.g. AMBR) that rely on context-sensitive reconstruction of old episodes performed in interaction with the mapping process.

Mapping

This is undoubtedly the core of analogy-making and therefore, all computational models of analogy-making include mapping mechanisms i.e. means of discovering which elements of the source correspond to which elements of the target. The difficulty is that one situation can be mapped onto a second situation in many different ways. We might, for example, make a mapping based on the color of the objects in both the source and target (the red-shirted individual in the base domain would be mapped to the red-shirted person in the target domain). This would, in general, be a very superficial mapping (but might, nonetheless, be appropriate on occasion). We could also map the objects in the two domains based on the relational structure. For example, we could decide that it was important to preserve the giver-receiver relationship in the first domain with the same relationship in the target domain, ignoring the fact that in the base domain the giver had a red shirt and in the target domain the receiver was wearing a red shirt.

Experimental work has demonstrated that finding this type of structural isomorphism between source and target domains is crucial for mapping [53]. Object similarity also plays a role in mapping, although generally a secondary one. A third factor is the pragmatic importance of various elements in the target – people try to find mappings that involve the most important elements in the target. Searching for the appropriate correspondences between the base and target is a computationally complex task that can lead to combinatorial explosion if the search is unconstrained.

Transfer of unmapped elements from source to target, thereby making inferences

This is the process of inserting new knowledge into the target domain based on the mapping. For example, assume a new type of car appears on the market and it turns out that this car maps well onto another type of car that is small, fast, and handles well on curves. But you also know that this latter type of car is frequently in need of repair. Transfer is when you wonder whether the analogous new model of car will also be in the garage often for repairs. Transfer is present in one form or another in most models of analogy-making and is typically integrated with mapping. Transfer is considered by some authors as an extension of the mapping already established, thus adding new elements to the target in such a way that the mapping can be extended.

Evaluation.

This is the process of establishing the likelihood that the transferred knowledge will turn out to be applicable to the target domain. In the example above, the evaluation process would have to assign the degree of confidence we would have that the new car would also frequently be in need of repair. Evaluation is often implicit in the mechanisms of mapping and transfer.

Only a few models of analogy-making have incorporated learning mechanisms, which is somewhat surprising since analogy-making is clearly a driving force behind much learning. However, some models are capable of generalization across the base and target, or across multiple exemplars, to form an abstract schema, as in LISA [54] and the SEQL model based on SME [55].

In the following we survey some specific models built on the above-mentioned contemporary works:

7.3.1 Classical Symbolic Models

ANALOGY

The earliest computational model of analogy-making, ANALOGY, was developed by Thomas Evans (1964). This program solves geometric-analogy problems of the form A:B::C:? taken from IQ tests and college entrance exams.

An important feature of this program is that the input is not a hand-coded, high-level description of the problem, but, rather, a low-level description of each component of the figure – dots, simple closed curves or polygons, and sets of closed curves or polygons. The program builds its own high-level representation describing the figures in A, B, C, and all given alternatives for the answer, with their properties and relationships (e.g. ((P1 P2 P3) . ((INSIDE P1 P2) (LEFT P1 P3) (LEFT P2 P3))). Then the program represents the relationship between A and B as a set of possible rules describing how figure A is transformed into figure B, e.g. ((MATCH P2 P4) (MATCH P1 P5) (REMOVE P3)) which means that the figure P2 from A corresponds to figure P4 from B, P1 to P5, and the figure P3 does not have a correspondent figure and is therefore deleted. Then each such rule is applied to C in order to get one of the alternative answers. In fact, each such rule would be

generalized in such a way to allow C to be applied to D. Finally, the most specific successful rule would be selected as an outcome. Arguably, one of the most significant aspects of the program is its ability to represent the target problem on its own — a feature that has unfortunately been dropped in most recent models.

Structure-Mapping Theory

Without question, the most influential family of computational models of analogy-making have been those based on Dedre Gentner's (1983) Structure Mapping Theory (SMT). This theory was the first to explicitly emphasize the importance of structural similarity between base and target domains, defined by common systems of relations between objects in the respective domains. Numerous psychological experiments have confirmed the crucial role of relational mappings in producing sound and convincing analogies. There are several important assumptions underlying the computational implementation of SMT called SME [55]: 1) mapping is largely isolated from other analogy-making sub-processes (such as representation, retrieval and evaluation) and is based on independent mechanisms; 2) relational matches are preferred over property matches; 3) only identical relations in both domains can be put into correspondence; 4) relations that are arguments of higher-order relations that can also be mapped have priority since they implement the "systematicity principle" that favors systems of relations over isolated relations; and 5) construction of two or three interpretations by a 'greedy merge' algorithm that generally finds the 'best' structurally coherent mapping. Early versions of SME mapped only identical relations and relied solely on relational structure. This purely structural approach was intended to ensure the domain-universal nature of the mapping process. Recent versions of SME have explored some limited use of pragmatic aspects of the situation, as well as re-representation techniques that allow initially non-matching predicates to match.

The MAC/FAC model [56] of analogical retrieval was developed to be coupled with SME. This model assumes that episodes are encapsulated representations of past events, which have a dual encoding in LTM: a detailed predicate-calculus representation of all the properties and relations of the objects in an episode and a shorter summary (a vector

representation indicating the relative frequency of predicates are used in the detailed representation). These representations are used in two sequential stages in the retrieval process. The first stage makes use of the vector representations to perform a superficial search for episodes that share predicates with the target problem. The episode vectors in LTM that are close to the target vector are then selected for processing by the second stage. The second stage uses the detailed predicate-calculus representations of the episodes to select the one that best matches the target. These two stages simulate the dominance of superficial similarity on retrieval, but also the fact that retrieval is influenced by the structural similarity.

The ideas of Gentner and colleagues, in particular, their emphasis on the structural aspects of analogical mappings, have been very influential in the area of analogy-making and have been applied to analogy-making in contexts ranging from child development to folk physics. Various improvements and variants of the SME have been developed over time and it has been included as a module in various practical applications.

Other Symbolic Models

A number of other symbolic models have played a role in the advance of analogy-making understanding. Jaime Carbonell proposed the concept of derivational analogy where the analogy is drawn not with the final solution of the old problem, but with its derivation, i.e. an analogy with the way of reaching up the solution is made, an approach developed further by Manuela Veloso. Smadar Kedar-Cabelli developed a model of purpose directed analogy-making in concept learning. Mark Burstein developed a model called CARL that learned from multiple analogies combining several bases. Mark Keane and his colleagues developed an incremental model of mapping, IAM, which would explain the order effects in presentation of the material. These symbolic models, as well as a number of other early symbolic models of analogy-making are described in detail in Hall (1989).

7.3.2 Connectionist Models

Research in the field of analogy-making has, until recently, been largely dominated by the symbolic approach for an obvious reason: symbolic models are well equipped to process and compare the complex structures required for analogy-making. In addition, in the early years of the new connectionist paradigm, these structures were very difficult to represent in a connectionist network. However, advances in connectionist representation techniques have allowed distributed connectionist models of analogy to be developed. Most importantly, distributed representations provide a natural internal measure of similarity, thereby allowing the system to handle the problem of similar, but not identical, relations in a relatively straightforward manner. This latter ability is crucial to analogy-making and has proved hard for symbolic models to implement.

Multiple Constraints Theory

The earliest attempt to design an architecture in which analogy-making was an emergent process of activation states of neuron-like objects was proposed by Keith Holyoak and Paul Thagard (1989) and implemented in a model called ACME. In this model, structural similarity, semantic similarity, and pragmatic importance determine a set of constraints to be simultaneously satisfied. The model is supplied with a representation of the target and one of the base and proceeds to build a localist constraint-satisfaction connectionist network where each node corresponds to a possible pairing hypothesis for each element of the base and each element of the target. So, for example, if the base is *train* and the target is *car*, then all elements of trains will be mapped to all elements of cars. There will therefore be hypothesis nodes created not only for "locomotive motor" but also for "locomotive → license plate," "locomotive → seat-belt buckle," etc. The excitatory and inhibitory links between these nodes implement the structural constraints. In this way, contradictory hypothesis nodes compete and do not become simultaneously active, while consistent ones mutually support each other. The network gradually reaches an equilibrium state and the best set of consistent mapping hypotheses (e.g., "locomotive" → "motor", "rails" → "road", etc.) wins. The relaxation of the network provides a parallel

evaluation of all possible mappings and finds the best one, which is represented by the set of most active hypothesis nodes. ARCS is a model of retrieval that is coupled with ACME and operates in a similar fashion. However, while mapping is dominated by structural similarity, retrieval is dominated by semantic similarity.

STAR

STAR-1 was the first distributed connectionist model of analogy-making (Halford, et al, 1994). It is based on the tensor product connectionist models developed by Smolensky. A proposition like MOTHER-OF (CAT, KITTEN) is represented by the tensor product of the three vectors corresponding to MOTHER-OF, CAT, and KITTEN: MOTHER-OF$\otimes$CAT$\otimes$KITTEN. The tensor product in this case is a three-dimensional array of numbers where the number in each cell is the multiplication of the three corresponding coordinates. This representation allows any of the arguments or the relational symbol to be extracted by a generalized dot-product operation: MOTHER-OF $\otimes$ CAT MOTHER -OF $\otimes$ CAT $\otimes$ KITTEN = KITTEN. The LTM of the system is represented by a tensor that is the sum of all tensor products representing the individual statements (the main restriction being that the propositions are simple and have the same number of arguments). Using this type of representation the model STAR-1 solves proportional analogies like CAT:KITTEN::MARE:?.

STAR-2 (Wilson, et al., 2001) maps complex analogies by sequentially focusing on various parts of the domains (simple propositions with no more than 4 dimensions) and finding the best map for the arguments of these propositions by parallel processing in the constraint satisfaction network (similarly to ACME). Since the number of units required for a tensor-product representation increases exponentially with the number of arguments of a predicate, this implies processing constraints in the model. The authors of the model claim that humans are subject to similar processing constraints, specifically, they can, in general, handle a maximum of four dimensions of a situation concurrently. The primary interest of the modelers is in exploring and explaining capacity limitations of human beings and achieving a better understanding of the development of analogy-making capabilities in

children.

LISA

John Hummel and Keith Holyoak (1997) proposed an alternative computational model of analogy-making using distributed representations of structure relying on dynamic binding. The idea is to introduce an explicit time axis so that patterns of activation can oscillate over time (thus the timing of activation becomes an additional parameter independent of the level of activation). In this way patterns of activation oscillating in synchrony are considered to be bound together while those oscillating out of synchrony are not. For example, "John hired Mary" requires synchronous oscillation of the patterns for "John" and "Employer" alternating it with synchronous oscillation of the patterns for "Mary" and "Employee". Alternating the activation of the two pairs periodically in time makes it possible to represent the whole statement. However, if the statement is too complex there will be too many pairs that need to fire in synchrony. Based on research on single cell recordings, Hummel and Holyoak believe that the number of such different pairs of synchronously firing concepts cannot exceed six. Representations in LISA's Working Memory are distributed over the network of semantic primitives, but are localist in Long Term Memory – there are separate units representing the episode, the propositions, their

subparts, predicates, arguments, and bindings. Retrieval is performed by spreading activation while mapping is performed by learning new connections between the most active nodes. LISA successfully integrates retrieval of a base with the mapping of the base and target, even though retrieval and mapping are still performed sequentially (mapping starts only after one episode is retrieved).

Hybrid Models

Two groups of researchers independently produced similar models of analogy-making based on the idea that high-level cognition emerges as a result of the continual interaction of relatively simple, low-level processing units, capable of doing only local computations. These models are a combination of both the symbolic and connectionist approaches.

Semantic knowledge is incorporated in order to compute the similarity between elements of both domains in a context-sensitive way.

COPYCAT, TABLETOP, etc.

The family of COPYCAT and TABLETOP architectures (Mitchell, 1993; Hofstadter, 1995; French, 1995) was explicitly designed to integrate top-down semantic information with bottom-up emergent processing. COPYCAT solves letter-string analogies of the form: ABC:ABD::KLM:? and gives probabilistically possible answers like KLN, KLD, etc. The architecture of COPYCAT involves a working memory, a semantic network (simulating long-term memory) defining the concepts used in the system and their relationships, and the Coderack – the procedural memory of the system – a store for small, nondeterministic computational agents ("codelets") working on the structures in the working memory and continually interacting with the semantic network. Codelets can build new structures or destroy old structures in working memory. The system gradually settles towards a set of consistent set of structures that will determine the mapping between the base and the target.

The most important feature of these models of analogy-making is their ability to build up their own representations of the problem, in contrast with most other models which receive the representations of the base and target as input. Thus these models abandon traditional sequential processing and allow representation-building and mapping to run in parallel and continually influence each other. In this way, the partial mapping can have an impact on further representation-building, thus allowing the gradual construction by the program of context-sensitive representations. In this way, the mapping may force us to see a situation from an unusual perspective in terms of another situation, this being crucial to creative analogy-making.

AMBR

AMBR (Kokinov, 1994) solves problems by analogy, e.g. "how can you heat some water in a wooden vessel being in the forest?". The solution, heating a knife in the fire and immersing it into the water, is found by analogy with boiling water in a glass using an

immersion heater.

The AMBR model is based on DUAL, a general cognitive architecture. The LTM of DUAL consist of many micro-agents each of which represents a small piece of knowledge. Thus concepts, instances and episodes are represented by (possibly overlapping) coalitions of micro- agents. Each micro-agent is hybrid – its symbolic part encodes the declarative and/or procedural knowledge it is representing, while its connectionist part computes the agent's activation level, which represents the relevance of this knowledge to the current context.

The symbolic processors run at speed proportional to their computed relevance thus making the behavior of the system highly context-sensitive. The AMBR model implements the interactive parallel work of recollection, mapping and transfer which emerge from the collective behavior of the agents and whose work produces the analogy. Recollection in AMBR-2 (Kokinov & Petrov, 2001) is reconstruction of the base episode in WM by activating relevant aspects of event information, of general knowledge, and of other episodes and forming a coherent representation, which will correspond to the target problem. The model predicts illusory memories, including insertions from general knowledge and blending with other episodes as well as context and priming effects. A number of these predictions have been experimentally confirmed.

The field of computer-modeling of analogy-making has moved from the early models which were intended mainly as existence proofs to demonstrate that computers could, in fact, be programmed to do analogy-making to complex models which make nontrivial predictions of human behavior. Researchers have come to appreciate the need for structural mapping of the source and target domains, for integration of and interaction between representation-building, retrieval, mapping and learning, and for building systems that can potentially scale up to the real world. Computational models of analogy-making have now been applied to a large number of cognitive domains (cf. Gentner, Holyoak, Kokinov, 2001). However, most of these models are severely constrained by the coverage, extensibility, and plausibility rather than sheer numeric possibility of expert and general knowledge they encode. In our approach, we show how we can easily integrate

structure-mapping into commonsense framework without being constrained severely by aforementioned limitations.

8 Discussions and Future Directions

i-Seek provides an exciting platform for expert advisory systems. Our effort heavily emphasizes the importance of providing means for *elicitation* and *explanation* in a mixed-initiative dialogue. Our research has shown that this framework combined with commonsense knowledge is very fruitful for the user in terms of providing expert advice to the user, making user learn by examples, and enhane the overall usability of the system. Nevertheless, it turns out that mapping from novice knowledge to expert knowledge and vice versa is not trivial. Firstly, natural language components are not very robust and accurate, which adds to some noise in the system behavior. As the natural language processing techniques advance, we would be able to provide more natural and unconstrained means of communication. Also, there are subtle issues related to optimality of common sense knowledge required to ascertain sufficiency for certain goal and how this can be characterized dynamically. Commonsense knowledge as it stands as of now has wide coverage, but at the same time makes way for very noisy and potentially contradictory knowledge. We believe that as this knowledge becomes more sanitized, the system would benefit hugely in making proper analogies. Our domain knowledge coverage is very sparse and as the project matures further, this knowledge would get richer and richer. In future, we plan to build knowledge acquisition tools, that any domain expert can use to input domain-specific knowledge. Besides, it would be interesting to see how this kind of interaction leads to social role building. As part of future activity, we aim to gather more domain-related common sense knowledge.

9 REFERENCES

1.Singh, P., Lin, T. Mueller, E., Lim, G. Perkins, T. and Li Zhu, W. Open Mind Common Sense: Knowledge acquisition from the general public. Proceedings of the First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems. (2000).

2.McCarthy, J., Programs with Common Sense, Teddington Conference on the Mechanization of Thought Processes (1959)

3.Personal Choice Point: Helping Users Visualize What It Means to Buy a BMW, Proceedings of the 2003 International Conference on Intelligent User Interfaces, January 2003, Miami, Florida

4.Timothy Chklovski. LEARNER: A System for Acquiring Commonsense Knowledge by Analogy. In Proceedings of Second International Conference on Knowledge Capture (K-CAP 2003). October 2003

5.W. Pratt, Yetisgen-Yildiz LitLinker: Capturing Connections across the Biomedical Literature, Proceedings of the International Conference on Knowledge Capture (K-Cap'03). Florida, October 2003.

6.Timothy Chklovski, Yolanda Gil, Varun Ratnakar, John Lee."TRELLIS: Supporting Decision Making via Argumentation in the Semantic Web", In Proceedings of 2nd International Semantic Web Conference (ISWC 2003), Web. (Short paper). 20-23 October 2003, Sanibel Island, Florida, USA

7.Ashwani Kumar, Sharad C. Sundararajan, Henry Lieberman, Common Sense Investing: Bridging gap between Expert and Novice, ACM Conference on Computer-Human Interface (CHI,2004), Vienna, Austria, April 2004;

8.Singh, P., Barry, B. and Liu,. H. Teaching machines about everyday life. BT Technology Journal, To Appear. Volume 22, forthcoming issue. Kluwer Academic Publishers. (2004).

9.Hugo Liu, Ted Selker, Henry Lieberman (2003). Visualizing the Affective Structure of a Text Document. Proceedings of the Conference on Human Factors in Computing Systems, CHI 2003, April 5-10, 2003, Ft. Lauderdale, FL, USA. ACM 2003, ISBN 1-58113-637-4, pp. 740-741.

10.Hugo Liu, Henry Lieberman, Ted Selker. (2002). GOOSE: A Goal-Oriented Search Engine With Commonsense. In De Bra, Brusilovsky, Conejo (Eds.): Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002, Malaga, Spain, May 29-31, 2002, Proceedings. Lecture Notes in Computer Science 2347 Springer 2002, ISBN 3-540-43737-1, pp. 253-263

11.Hugo Liu and Henry Lieberman (2002). Robust photo retrieval using world semantics. In Proceedings of the LREC 2002 Workshop on Creating and Using Semantics for Information Retrieval and Filtering: State-of-the-art and Future Research, Las Palmas, Canary Islands, pp. 15-20.

12.Nathan Eagle and Push Singh (2004). Context sensing using speech and common sense. To appear in Proceedings of the NAACL/HLT 2004 workshop on Higher-Level Linguistic and Other Knowledge for Automatic Speech Processing.

13.Rauterberg, M. (1994). Human information processing in man-machine interaction. In: A. Grieco, G. Molteni, E. Occhipinti & B. Piccoli (eds.), Book of Short Papers of 4th International

Conference on Work with Display Units--WWDU'94 (Volume 2, pp. E68-E71). University of Milan: Institute of Occupational Health.

14. Guiding people to information: providing an interface to a digital library using reference as a basis for indexing (2000), Shannon Bradshaw, Andrei Scheinkman, Kristian Hammond, Intelligent User Interfaces

15. Liu, H. & Singh, P. ConceptNet: A Practical Commonsense Reasoning Toolkit. BT Technology Journal, To Appear. Volume 22, forthcoming issue. Kluwer Academic Publishers. (2004).

16. Singh, P., and Williams, W. LifeNet: a propositional model of ordinary human activity. Proceedings of the Workshop on Distributed and Collaborative Knowledge Capture (DC-KCAP) at K-CAP (2003).

17. Hugo Liu, Henry Lieberman, Ted Selker. (2002). GOOSE: A Goal-Oriented Search Engine with Commonsense. In De Bra, Brusilovsky, Conejo (Eds.): Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002, Malaga, Spain, May 29-31, 2002, Proceedings. Lecture Notes in Computer Science 2347 Springer 2002, ISBN 3-540-43737-1, pp. 253-263

18. Henry Lieberman and Barbara Barry, Collecting commonsense experiences. 2003. Proceedings of the Second International Conference on Knowledge Capture (K-CAP 2003). Florida, USA

19. Plan-based interfaces: keeping track of user tasks and acting to cooperate, (2002) David Franklin Jay Budzik , Kristian Hammond International Conference on Intelligent User Interfaces

20. High-Level Perception, Representation,and Analogy:A Critique of Artificial Intelligence MethodologyDavid J. Chalmers, Robert M. French, Douglas R. Hofstadter CRCC Technical Report 49, March 1991

21. User interfaces with semi-formal representations: a study of designing argumentation structures, Proceedings of the 10th international conference on Intelligent user interfaces table of contents, San Diego, California, USA, Timothy Chklovski   University of Southern California, Marina del Rey, CA, Varun Ratnakar   University of Southern California, Marina del Rey, CA, Yolanda Gil

22. Wang, D., Kaufman, D.R., Mendonca, E.A., Seol, Y.H. Johnson, S.B., & Cimino, J.J. (in press). The Cognitive Demands of an Innovative Query User Interface. In the Proceedings of the American Medical Informatics Annual Fall Symposium. Philadelphia: Hanley & Belfus.

23. Gentner, D., & Markman, A.B. (1997). Structure mapping in analogy and similarity. American Psychologist, 52(1), 45-56.

24. http: //www.vanguard.com

25. http://www.fool.com

26. http://www.nasdaq.com

27. http://www.morningstar.com

28. http://www.investopedia.com

29. http://finance.yahoo.com

30. http://www.cnn.com/money

31. End-User Debugging for Electronic Commerce, (with Earl Wagner) ACM Conference on Intelligent User Interfaces, Miami Beach, January 2003.

32. Ariely, Dan, George Loewenstein, and Drazen Prelec (2005), "Tom Sawyer and the Myth of Fundamental Value," Journal of Economic Behavior and Organization, forthcoming.

33. Thaler, R. (1999). "Mental Accounting Matters." Journal of Behavioral Decision Making 12: 183 - 206.

34. Save More Tomorrow: Using Behavioral Economics to Increafconceptnetse Employee Saving, SHLOMO BENARTZI, Journal of Political Economy, Vol. 112, No. 1, pp. S164-S187, February 2004

35. Shafir, Eldar, Peter Diamond, and Amos Tversky, "Money Illusion," Quarterly Journal of Economics, 112(2): 341–374, May 1997

36. Fillmore, Charles J. (1968): The case for case. In Bach and Harms (Ed.): Universals in Linguistic Theory (pp. 1-88), Holt, Rinehart, and Winston, New York.

37. Fillmore, Charles J. (1976): Frame semantics and the nature of language. In Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech, Volume 280 (pp. 20-32).

38. Fillmore, Charles J. (1982): Frame semantics. In Linguistics in the Morning Calm (pp. 111-137), Hanshin Publishing Co., Seoul, South Korea.

39. Fillmore, Charles J. (1985): Frames and the semantics of understanding. In Quaderni di Semantica, Vol 6, No. 2 (pp. 222-254).

40. Timothy Chklovski. LEARNER: A System for Acquiring Commonsense Knowledge by Analogy. In Proceedings of Second International Conference on Knowledge Capture (K-CAP 2003). October 2003.

41. Kai Chang, P. Raman, W. Carlisle, and J. Cross, "A Self-improving Helpdesk Service System Using Case-Based Reasoning Techniques," Computers in Industry, Vol. 30, 1996, pp. 113-115.

42. William R. Swartout and Yolanda Gil. "EXPECT: A User-Centered Environment for the Development and Adaptation of Knowledge-Based Planning Aids". In Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative, ed. Austin Tate. Menlo Park, Calif.: AAAI Press, 1996

43. "Knowledge Mobility: Semantics for the Web as a White Knight for Knowledge-Based Systems" , Yolanda Gil, In "Developing the Semantic Web", D.Fensel, J. Hendler, H. Lieberman, W. Whalster(Eds), MIT Press, forthcoming

44. Burns, H. L. and C. G. Capps. (1988) "Foundations of Intelligent Tutoring Systems: An Introduction," Foundations of Intelligent Tutoring Systems, Lawrence Erlbaum Associates, Hillsdale, NJ.Systems Conference, Washington, D. C.

45. Horowitz, B. M. (1991) Importance of Architecture in DOD Software," MITRE Technical Paper, M91-35.

46. Massey, D., L. Kurland, Y. Tenney, J. de Bruin, and K. C. Quayle. (December 1986) HAWK MACH-III, Intelligent Maintenance Tutor Design Development Report, ARI Research Note 86-99, Training Research Laboratory, U. S. Army, Research Institute for the Behavioral and Socail Sciences.

47. Gott, S. P. and R. Pokorny. (December 1987) "The Training of Experts for High-Tech Work Environments," 9th Interservice/Industry Training

48. Gentner, D., Holyoak, K. J., & Kokinov, B. (Eds.). (2001). The analogical mind: Perspectives from cognitive science. Cambridge, MA: MIT Press.

49. Mitchell, M. (1993) Analogy-making as Perception: A computer model. Cambridge, MA: MIT, Press

50. Hofstadter, D. and the FARG (1995). Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought. NY: Basic Book

51. French, R. (1995). The Subtlety of Sameness: A theory and computer model of analogy-making.Cambridge, MA: MIT Press

52. Kokinov, B., Petrov, A. (2001). Integration of Memory and Reasoning in Analogy-Making: The AMBR Model. In: Gentner, D., Holyoak, K., Kokinov, B. (eds.) *The Analogical Mind*, Cambridge, MA: MIT Press

53. Gentner, D. (1983). Structure-Mapping: A Theoretical Framework for Analogy, *CognitiveScience*, 7(2), 155-170

54. Hummel, J. & Holyoak, K. (1997). Distributed Representations of Structure: A Theory ofAnalogical Access and Mapping. Psychological Review, 104, 427-466.

55. Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence, 41,* 1-63

56. Forbus, K., Gentner, D. & Law, K. (1995). MAC/FAC: A Model of Similarity-Based Retrieval.*Cognitive Science*, vol. 19 (2)