Providing Expert Advice by Analogy for On-Line Help

Henry Lieberman and Ashwani Kumar
Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA
{lieber, ashwani}@media.mit.edu

Abstract

One of the principal problems of online help is the mismatch between the specialized knowledge and technical vocabulary of experts who are providing the help, and the relative naïveté of novices, who usually are often not in a position to understand solutions expressed by the expert in their own terms.

Most of the interfaces are plagued by recurrent key problems: 1) elicitation – how to ask questions that enable the helper 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. One of the best ways to do this is for the expert to provide analogies in terms of Commonsense knowledge, which provide metaphors that help novices learn problem-solving skills.

SuggestDesk is a system that acts as an advisor to an online technical support person. It uses a large Commonsense knowledge base to search for analogies between known technical problem-solution pairs, and situations and events in everyday life that can be used to explain them.

Introduction

The success of Web interactions is becoming increasingly dependent on online help and online technical support. If Web interactions are to increasingly replace brick-and-mortar physical interactions, users must have confidence that any problems that might arise can be effectively dealt with. Static documentation on stored HTML pages no longer suffices in many cases. In simple cases, automated advice programs hold potential for being able to automatically sense the user's context and perform some solution steps automatically. But for more difficult situations, many problems must still be referred to human online assistants, in real-time text chat, or in telephone conversations where both parties are interacting with Web-based systems.

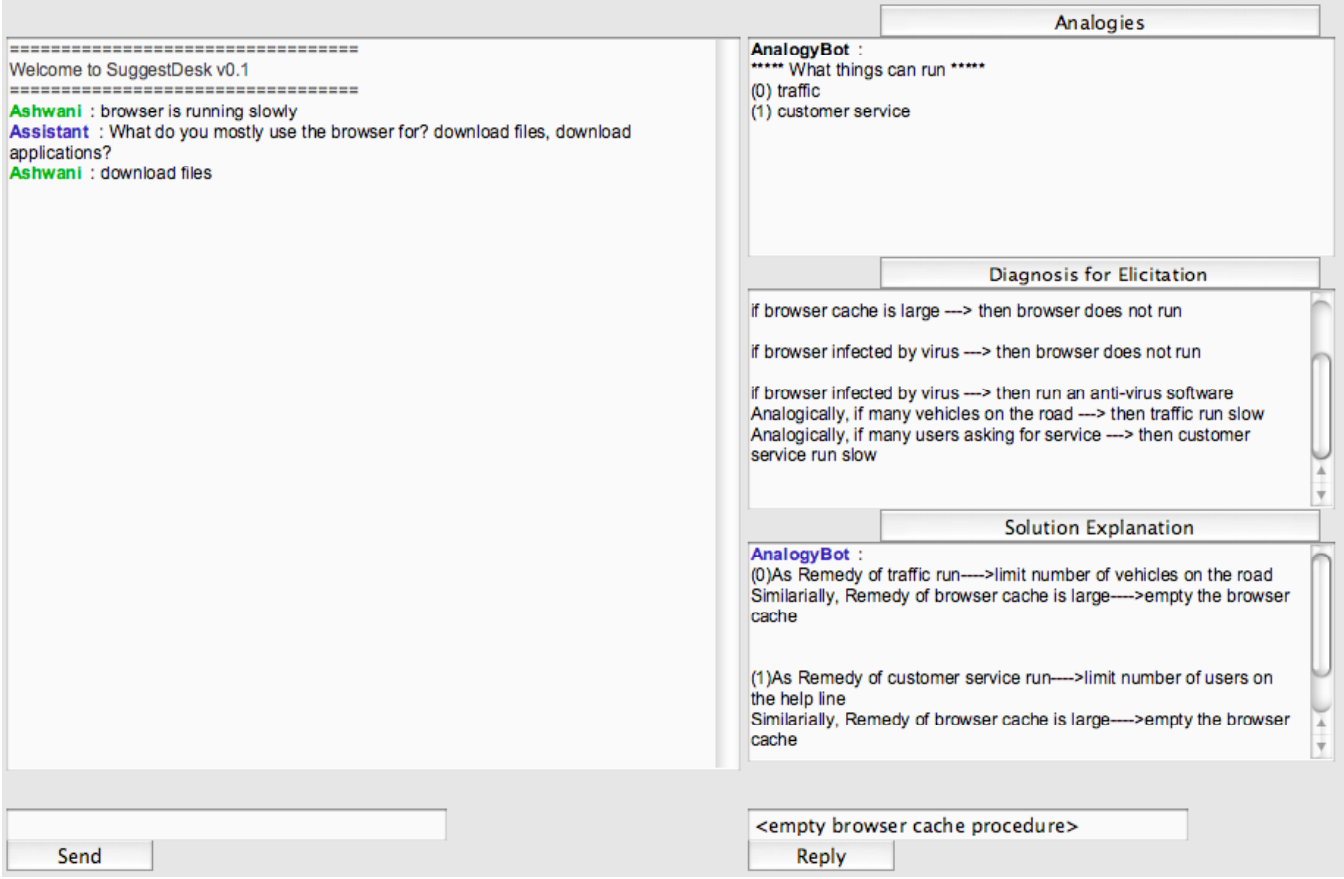
Many current support personnel are aided by problem-solution databases. These store associations between anticipated or experienced problems, and known solutions. A common behavior of a support person is to lookup the user's problem in the database. When a good match is found, the support person can simply report the solution to the user. More sophisticated systems use Case-Based Reasoning [3], which adds the ability to do approximate matching, and appropriate modification of the solution to particular circumstances.

That all works well, providing the expert and the user share enough vocabulary and technical expertise to express the problem and understand the solution in the way it is described. But novices often do not have the technical vocabulary nor specific expertise needed to understand the expert's questions about the problem and the expert's exposition of what to do. What then?

Traditional Expert Systems and Case-Based Reasoning systems are good at encoding expert knowledge. But they are not so good at modeling the knowledge of the novice. At best, those systems that do have some model of the novice's knowledge are limited to saying what subset of the expert knowledge is expected to be shared with the novice. This is always limited to the narrow technical domain in which the expert's knowledge resides. For example, a model might say that the average user can be expected to know how to double-click, but might not know what encryption is.

We believe that the best elicitation and explanation strategies involve the expert trying to make an analogy between technical aspects of the problem and solution, and the user's experience of everyday life, even if the analogies may be to domains outside of a technical realm. To have the machine assist in doing this, we need a model of everyday life.

We have a unique resource in Open Mind Common Sense [OMCS], a knowledge base containing 750,000 English sentences obtained from volunteer contributors [8], and its derivative knowledge bases ConceptNet [9] and LifeNet. We also have a limited ability to perform analogies between Commonsense concepts in the ConceptNet semantic network.



We have implemented a computer assistant for a human acting as a technical support person for a novice user, and communicating with that user via typed chat, called *SuggestDesk*. We assume SuggestDesk is also being used in conjunction with a conventional problem-solution database. The purpose of SuggestDesk is to watch the interaction between the novice and the helper, and to suggest to the helper analogies that will help him or her elicit problem information from the user and explain technical solutions.

We do not attempt to have the computer agent interact directly with the end-user, because our capability for understanding natural language and producing analogies is limited. The hope is that SuggestDesk will occasionally suggest interesting analogies to the helper that will aid them in their conversations with the end user.

Knowledge Engineering With Technical Support People

To better understand how technical support people achieve success in helping users with problems, we interviewed members of the online technical support team at America Online. We asked for examples where experts were able to make complex technical solutions understandable to novices. We received many good examples of such interactions, involving making analogies between technical problems and common situations in everday life. An example follows.

In response to a user reporting an inability to access secure Web sites, the online technical support helper looked up the problem in a conventional problem-solution database and retrieved the following solution procedure.

Solution Procedure: "Check if cipher strength is '0' Upgrade Browser to 128 bit Encryption."

Obviously, many AOL users, who are often inexperienced in technical aspects of computer use, would have difficulty understanding what this solution means, why it would work, and even what problem it was trying to solve. (The first author, who holds a doctoral degree and has more than 30 years of computer experience, admits that he doesn't understand it, either).

There are two ways in which the helper provided useful explanation. The first is in terms of the technical aspects of the problem.

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.

This explanation, though correct, might be unintelligible to a user with only limited computer experience.

But the helper didn't stop there. The second part provides an analogy that helps give the user not only the reason for the solution, but provides an analogy to an everyday situation the user is likely to be familiar with.

Analogy: If 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."

Analogies help the user learn the relation between important concepts and aspects of the technical solution ("Oh, I guess that means '128-bit encryption' must be like some sort of security clearance."). Even if this analogy is not perfect, it gives the user some skills that can be applied to other similar problems, and possible inaccuracies can later be refined.

The SuggestDesk Application

The SuggestDesk application illustrated above implements an interactive chat-based client, which both the user and the Help Assistant use. The interface 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 everyday 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.

The Open Mind Common Sense knowledge base

Since the fall of 2000 the MIT Media Lab has been collecting commonsense facts from the general public through a Web site called Open Mind Common Sense [OMCS], which you can find at **openmind.media.mit.edu**. At the time of this writing, the Open Mind Common Sense Project has collected over 750,000 facts from over 16,000 participants. These facts are submitted by users as natural language statements of the form "tennis is a sport" and "playing tennis requires a tennis racket." While Open Mind does not contain a complete set of all the common sense knowledge found in the world, its knowledge base is sufficiently large enough to be useful in real world applications.

Using natural language processing, the Open Mind knowledge base was mined to create ConceptNet [9], a large-scale semantic network currently containing over 300,000 Commonsense concepts. ConceptNet consists of logical predicates of the form: [IsA "tennis" "sport"] and [EventForGoalEvent "play tennis" "have racket"]. ConceptNet is similar to WordNet [4] in that it is a large semantic network of concepts, however ConceptNet contains everyday knowledge about the world, while WordNet follows a more formal and taxonomic structure. For instance, WordNet would identify a *dog* as a type of *canine*, which is a type of *carnivore*, which is a kind of *placental mammal*. ConceptNet identifies a *dog* as a type of *pet,* its most salient feature for Commonsense reasoning.

ConceptNet has some limited ability to do analogical reasoning. Its idea of analogy is to generalize concepts according to their participation in playing a role in some

piece of Commonsense knowledge. For example, ConceptNet can propose *love* as an analogy to *money*, if it has *"People will do anything for love"*, and *"People would do anything for money"*.

OMCS constitutes, in some sense, a *generic novice model*. It represents what the average user can be "expected to know", unless you know otherwise. Therefore we can reduce the problem of finding an analogy to a technical situation that a novice can understand, to the problem of finding an analogy between the technical concepts and concepts that appear in ConceptNet.

An example: Why does my browser run slowly?

In the example screen in the first illustration, the user complained that their browser is running slowly. We ask ConceptNet, *"What can run slow?"* and we get *"Traffic can run slow"* and *"Customer service can be slow"*.

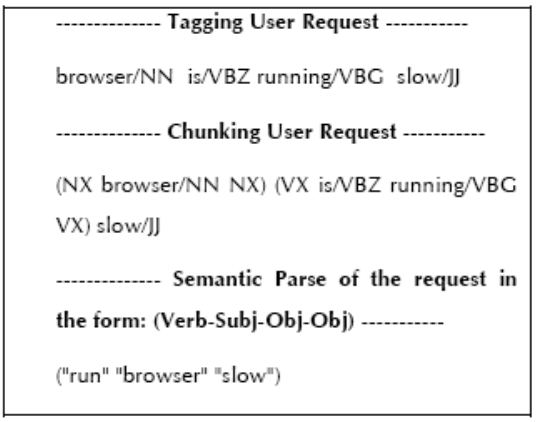
These provide fodder for analogies. Traffic runs slow because there are too many cars on the road. What is analogous to a car in the case of a browser running slow? Details are filled in using a method similar to Gentner's Structure Mapping Engine [5]. Too many users using the AOL service at the same time is analogous to too many cars on a road at rush hour. What is the solution? In the case of traffic, try to travel at a time other than rush hour. In the case of congestion on an on-line service, try to log in at a later time.

Another possibility is that the browser is infected by a virus. In order to determine whether this is the case, the helper must *elicit* details of the user's situation. Since computer viruses are often contracted by downloading applications, the helper asks the user if they have been recently downloading new applications. If this is the case, the helper can explain the effects of a computer virus by making an analogy to a biological virus. *"You know how when you have the flu, you can't do things as fast as you normally do?"*.

Implementation of SuggestDesk

The system performs Natural Language analysis with Hugo Liu's MontyLingua Part-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.

Below, the 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, which is then correlated with the lexical predicates in ConceptNet.

<<<Frame Name: run >>>
Type : event
Subject : browser
Modifier: slow
Objects: <>

After the natural language analysis, the major components are

• The Commonsense Processor (CP),
• The Expert Analyzer (EA),
• The Analogy Mapping Engine (AME), and
• The Elicitation and Explanation Processor (EEP).

The Commonsense Processor and Expert Analyzer work similarly, processing natural language utterances and producing semantic networks. The EA uses the AOL Help knowledge base to mine help topics related to key concepts in the help domain, such as the following:

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 structures are organized into a semantic graph called *ExpertNet*, where nodes represent domain-specific concepts and edges represent the relations. For instance, ExpertNet has the following structures related to Internet and browsers:

(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 'PC infected by virus' 'browser run slow')

The Analogy Mapping Engine (AME) uses ConceptNet and ExpertNet as constructed above to perform novice-expert knowledge mapping. Since both ConceptNet and ExpertNet are similar in graph structure, the AME is able to perform a 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, 1.1887218755408673], [CapableOfReceivingAction, crash, 1.1887218755408673], [CapableOfReceivingAction, start, 1.1887218755408673]], 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 *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 and i = incoming edges

The Elicitation and Explanation Processor (EEP) retrieves analogies from AME and processes the ranked list of analogies that match the given user's 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. After retrieving the likely analogous causes, it outputs the analogies and associated diagnoses in the "Diagnosis for Elicitation" window. Thus, EEP enables delivery of analogies and associated diagnostic information to the SuggestDesk UI.

Evaluation

We performed a small experiment comparing SuggestDesk with the existing AOL HelpDesk system, with 1 AOL collaborator and 5 MIT students as subjects. We measured task execution time, success rates, and subjective satisfaction on two problems, one related to browser performance issue, and the other one related to a 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 SuggestDesk 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 tasks, similar in nature to the experimental tasks. For the subjective satisfaction variable, users filled out a user satisfaction questionnaire upon completion of the experiment.

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

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

As the results indicate in the above two tables, on average 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.

Related Work

Our research touches various aspects of contemporary research in Artificial Intelligence. We don't have time and space to cover all related subjects, but we will touch upon three major areas: Knowledge Acquisition, Intelligent Tutoring Systems, and Analogy.

The most direct is Knowledge Acquisition, though that addresses primarily elicitation and not explanation. We would like to particularly acknowledge the work at USC/ISI on the EXPECT and TRELLIS systems [1, 6, 11]. EXPECT 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 application for argumentation and decision-making. The system supports the user in creating knowledge snippets from online resources. The key is to capture how the user progressively generates new knowledge that results in added value to the original raw information sources. EXPECT and TRELLIS assume that the user and the system are roughly equally expert and share vocabulary and knowledge.

Conversely, Intelligent Tutoring Systems focus on explanation rather than elicitation. Many have student models as well as models of the expert knowledge to be taught, but again, limited to the particular domain rather than general Commonsense knowledge. [2] is a good general overview of the field.

Finally, there has been much study of analogy. Gentner's Structure-Mapping Engine [5] is the classic reference, and the basis for our expert/novice knowledge alignment. Liu's ConceptNet paper describes in detail ConceptNet's analogy features. We should also point out that in addition to the symbolic model of Gentner, connectionist models such as Mitchell and Hofstadter's CopyCat [10] are also important. Our model falls somewhere in between, as ConceptNet's spreading activation is somewhat connectionist in nature whereas Gentner's Structure Mapping is used to fill in roles.

We have also previously worked in the area of online help, on the problem of how to choose which helper or peer might best have knowledge about a particular user's specific problem [12]. We have also created systems for tracking Web procedures performed by a user and visualizing explanations of them and providing debugging tools for self-help [13].

Acknowledgements

We would like to thank Amy Hale and Tom Jarmolowski of America Online, as well as several members of the AOL Help team, for providing financial support, knowledge engineering help and general support and advice.

References

1. Jim Blythe, Jihie Kim, Surya Ramchandran, and Yolanda Gil, An Integrated Environment for Knowledge Acquistion, Proceeding of the 2001 International conference on Intelligent User Interfaces (IUI-2001), Santa Fe, New Mexico, January 2001.
2. H. L. Burns, and C. G. Capps. (1988) "Foundations of Intelligent Tutoring Systems: An Introduction," Foundations of Intelligent Tutoring Systems, Lawrence Erlbaum Associates, Hillsdale, NJ.
3. 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.
4. C. Fellbaum, WordNet: An electronic lexical database. MIT Press, Cambridge, MA, USA, (1998).
5. D. Gentner, (1983). Structure-Mapping: A Theoretical Framework for Analogy, CognitiveScience, 7(2), 155-170
6. Yolanda Gil, Knowledge Mobility: Semantics for the Web as a White Knight for Knowledge-Based Systems", In "Spinning the Semantic Web", D.Fensel, J. Hendler, H. Lieberman, W. Wahlster (Eds), MIT Press, 2003.

7. Ashwani Kumar, Sharad C. Sundararajan, Henry Lieberman, Common Sense Investing: Bridging the gap between Expert and Novice, ACM Conference on Computer-Human Interface (CHI,2004), Vienna, Austria, April 2004.
8. H. Lieberman, H. Liu, P. Singh, B. Barry, Beating Common Sense into Interactive Applications, *AI Magazine*, Winter 2005.
9. H. Liu, & P. Singh, ConceptNet: A Practical Commonsense Reasoning Toolkit. British Telecom Technology Journal, Volume 22, No. 4, Kluwer Academic Publishers. (October 2004).
10. M. Mitchell, (1993) Analogy-making as Perception: A computer model. Cambridge, MA: MIT Press.
11. 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/
12. A. Vivacqua and H. Lieberman, Agents to Assist in Finding Help. ACM Conference on Computers and Human Interface. CHI-2000.
13. E. Wagner and H. Lieberman, Supporting User Hypotheses in Problem Diagnosis on the Web and Elsewhere, ACM Conference on Intelligent User Interfaces, Funchal, Madeira, Portugal, January 2004.