ABSTRACT
In this paper, we describe a way to understand natural language by partial processing. More specifically, the aim is to infer a user goal from a submitted query or an ongoing dialogue. The system is a part of the museum guide robot project Isaac, which was finished in June 1999.

KEYWORDS
Natural language understanding, dialogue models, inferring user goals

INTRODUCTION
A problem in human-computer interaction today is that natural language can not be used in most situations where it would be feasible. With the advent of speech recognition software, the situation is starting to change. However, these products interpret natural speech to text, and an analysis of the text is needed to infer the user’s goals and plans. Isaac is using both speech recognition and speech synthesis to communicate with the user. The language engine processes an input query string, and produces an action that meets the user’s goal.

ISAAC
Isaac is an autonomous museum guide robot, able to navigate in a museum environment, and communicate with museum visitors in natural language. The language module of the system uses several steps and techniques to achieve the user’s goal. The language engine described takes textual input, where a single dialogue at a time is considered. The user is supposed to have one clear and concise goal in mind, alternatively a plan to reach an unknown goal, and that this goal or plan is submitted through the input string. References to previous dialogues or objects in the surroundings are accepted. The result of the processing is an answer generation, which is presented to the user in spoken voice. Isaac also takes initiative and produces active help. The system was implemented in 1999 as a project of ITP at Uppsala University. In the following sections, Isaac’s view of human language is presented. We then continue with an explanation of how Isaac actually carries out the mappings between the user’s query and Isaac’s internal representation of human language.

After this (and maybe a cup of coffee), the adaptive nature of our method will be examined, leading to an algorithm for adapting parts of the internal knowledge base that is the key for Isaac’s ability to understand human language.

PARTIAL ADAPTIVE TERM SPOTTING
Isaac follows a grammar different from the regular linguistic grammar. It consists of the following word classes:

- Object
- Subject
- Attribute
- Command
- Reference
- Name
- Emotion
- Greeting

These classes have the relations described below.

- Objects are defined by their attributes. An object can have any number of attributes (1:N relation).
- Subjects are actors that have emotions about objects. Subjects can have any number of emotions about any number of objects. (1:N:N)
- Commands operate on objects. (1:1)
- References are operators that define a group of objects. (1:N)
- Names are operators that define individual objects. (N:1)
- Emotions are inexact definitions of objects that subjects carry out. Emotions may vary over time.

To understand the user’s goal, Isaac must translate the query given by the user. Such a query consists of words, or terms, as we shall label them henceforth. A query generally follows the English grammar, since this is the user’s usual way of communicating. We do not distinguish between spoken and written grammar; even though both spoken and written language are accepted [5]. However, Isaac is not aware of the rules that make up this grammar – instead he uses a simplified set of rules that comprise the basic patterns of a sentence.
A simplified view of human language

In the system's view, there are three important parts, or functions, of a given sentence. This is an oversimplified view of human language, but we claim that it is an adequate approach in the specific case of a museum scenario.

![Diagram showing Subject, Object, Object Agent]

The subject is the conventional subject in human language and traditional linguistics. The subject is obtained from the dialogue history, which loosely corresponds to the human short time memory. Objects also correspond to its relative in human language. An object agent may be thought of as a word that cannot stand on its own. It acts on, defines, or relates an object. Object agents are made up of four subclasses, pictured below.

![Diagram showing Attribute, Command, Emotion, Greeting]

Attributes define objects. For example, a painting may have the attribute *age*. A command is exactly what it sounds like. Examples of commands are *go to,* and *stop.* Emotions and greetings are self-explanatory. An emotion is always expressed by a subject about an object. Greetings also imply a subject and an object but these rarely show up explicitly in a sentence.

An object is characterized by its attributes and especially important are *names.* As we shall see, our implementation uses names and the final word class, *references,* to decide which object the user refers to. References are words that put conditions on objects. For instance, one of the most usual references is *this.* The word *this* implies that the object is already mentioned, or that it is physically near the point where Isaac and the user are located.

In conclusion, references narrow the group of suitable object candidates to those which fulfill certain conditions, while explicit names usually narrow it down to a very small set of candidates. The transition goes directly from Reference to an instance of a Name. A few examples that explain the different functions and classes mentioned above:

<table>
<thead>
<tr>
<th>Subject</th>
<th>Object Agent</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>like</td>
<td>Mona Lisa</td>
</tr>
<tr>
<td>Who made</td>
<td>it?</td>
<td></td>
</tr>
<tr>
<td>Go to</td>
<td>the exit!</td>
<td></td>
</tr>
</tbody>
</table>

Mapping of text to internal representation

With the foundation laid in the last section we now turn our attention to how the actual mappings between a user query and the system's internal representation of human language are carried out.

We have based our algorithms on the assumption that human users often do not use perfect English. Words may be left out for example, misspellings are likely to occur, and the speech-to-text mappings are error-prone. In addition, a query may very well be ambiguous, making our task of understanding it even harder. For these reasons, we have devised a model that is based on partial understanding and finding a set of possible results (user goals). This set is ranked according to the estimated probability that the query indeed does originate from the particular user goal. Hence, we pick the user goal that has obtained the largest probability. We may also combine several goals that are equally likely, in order to obtain a better answer.

Every instance of a word class (an example may be; *Age* is an instance of the class *Attribute*) has a set of words associated with it. A certain word may have a stronger association with a given instance than another. For this reason, the associations are weighted. The higher its weight, the more probable is it that the word actually implies the given instance. The set of associations is dynamic by its nature. This quality will be investigated further in the next section.

The estimation of probabilities is carried out by searching the user query in three steps:

1. Find explicitly named objects.
2. Find references.
3. Find object agents.

The results from these searches are ranking lists of probable instances of the searched class. Together, these lists are used to create a ranking list of likely user goals. Each of the mentioned steps is carried out in a similar manner.

1. For each word in the query: Is there an association with any instance of the particular word class? Increase its probability (we may call this a weight too) by the associated weight.
2. Sort the resulting list according to the entries' total weights.

When all three phases have been carried out, the lists are merged into a new list that is a representation of the user's goal as understood by Isaac. From this list and knowledge about legal combinations of instances in the lists, we are able to estimate the user's goal and produce an answer tuple accordingly.

As the avid reader may notice, subjects are not considered in the implemented system. Why is this? Actually, it turns out that in the particular scenario, the subject is either Isaac or the user. Whom of these, was rarely important to realize. In those cases where it was needed, our simple dialogue
state/model was used instead of actual parsing (although, to be correct, the dialogue state depends on the understanding of queries).

ADAPTATION

The mappings of terms to different elements within the word classes should be easy to change. We want a system that is able to learn from its mistakes, and thus we use a simple adaptive scheme in order to train Isaac’s answering abilities. In short, trusted users are allowed to query Isaac and, if the answer is not considered appropriate, to tell Isaac what the answer should have been (out of a list of possible alternatives). Then, Isaac’s mappings are updated by a simple set of equations. Essentially, we must state how much the total sum of the query should be for the correct alternative. This is easily calculated since we know the scores of the competing alternatives. After this, we must distribute the total increment over the words that made up the query. We chose to give a higher increase to such words that are not affecting many user goals. The objective is to minimize the damage for other queries. This process is derived from adaptive signal processing, in contrast to the alternative (non-adaptive) solution based upon a Bayesian network presented by Horwitz [2]. One benefit from our approach is that the learning process continues over time of usage.

REFERENCE STACK

References to previous dialogues are supported through a reference stack. All objects talked about are thrown on the stack, as well as objects that are tightly related. If a painting is mentioned, the painting’s artist is also thrown. Museum objects around the robot are also present on the stack, as well as objects Isaac is always aware about (like Isaac himself). Each object on the stack is interpreted to legal references when a reference is made. If the artist is a man, only “he” and similar references are legal. The stack is also time-dependent: objects disappear after a while. All objects on the stack are mirrored in the graphic user interface.

DIALOGUE MODEL

The dialogue model used can be separated into three parts. The following dialogue is authentic from the system evaluation. U is User and I Isaac.

The first part is simple dialogue or simple dialogue with references. Here Isaac responds to a simple command, and answers a question containing a reference to previous dialogue.

<table>
<thead>
<tr>
<th>Speaker : Query</th>
<th>Dialogue State</th>
</tr>
</thead>
<tbody>
<tr>
<td>U: Go to Mona Lisa</td>
<td>Command</td>
</tr>
<tr>
<td>I: Follow me to Mona Lisa</td>
<td>Confirmation</td>
</tr>
<tr>
<td>U: Who painted it? (reference)</td>
<td>Question</td>
</tr>
<tr>
<td>I: Da Vinci painted Mona Lisa</td>
<td>Answer</td>
</tr>
</tbody>
</table>

The second type of dialogue model is mixed interaction with initiative. Usually, the user is the one taking initiative. But Isaac uses the advantages of taking initiative when a particularly hard question is asked.

U: Do you like it? Emotional question
I: Do you? Counter question
U: Yes. Response
I: Then you might like the Last Supper. Initiative

The third instance of the dialogue model is predictive active help. If a users is asking a completely out of order question, Isaac understands that the question is illegal, and corresponds with providing some information of objects in the immediate surrounding.

U: Why are the leaves green? Illegal question
I: Ask me about Venus From Mars instead. Active help

The dialogue model helps Isaac to identify in which state the dialogue is at every time, which makes the dialogue more precise. The dialogue history forms the user model.

INTERFACE DESIGN AND ACTION GENERATION

There are several user interfaces in this system, supporting spoken and textual input as well as spoken and graphical output. The visual feedback shows the dialogue history, and reflects the objects around in an environment mirror. It also shows Isaac’s current emotional state by a simple face. The mood transition goes smoothly from sad (one extreme) too happy (the other extreme); Isaac is easy to cheer up by asking a well formulated question. Isaac gets annoyed when a lot of irrelevant or non-understandable questions are asked.

Happy       Flattering    Indifferent       Angry  Sad

The spoken response takes into consideration question understanding (I don’t understand), question quality (I don’t know when Mona Lisa was born), dialogue history (As I just told you...), emotions (Are you speaking Japanese?) and flatter (By the way, I like your haircut). By using these techniques, the user is manipulated to believe that the system is more intelligent than it really is.

DISCUSSION

The small-scale prototype Isaac could possibly be scaled to a larger information space. The language engine is designed
to be portable. Today, natural speech understanding is not
100% (IBM ViaVoice was used in Isaac), which resulted in
a less precise language understanding. The dialogue
produced by the system is fast, it seems intelligent, and
most of the questions are answered correctly.

ACKNOWLEDGMENTS
We thank IT95 for the splendid cooperation and Jussi
Karlgren for ideas.

REFERENCES
2. E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K.
Rommelse. The Lumiere Project: Bayesian User
Modeling for Inferring the Goals and Needs of Software
Users. Proceedings of the Fourteenth Conference on
Uncertainty in Artificial Intelligence, Madison, WI, July
3. S. Thrun, M. Bennewitz, W. Burgard, F. Dellaert, D.
Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and
D. Schulz. MINERVA: A second generation mobile
tour-guide robot. 1997
4. I.R. Nourbakhsh. The failures of a self-reliant tour
robot with no planner. 1998.
5. ITP, Uppsala University. Isaac - a Tour Guiding
System. 1999. www.docs.uu.se/isaac
6. Telemann. Har tal och skriftspråk olika grammatiker?
1996
1985.