

A light green silhouette of a person from the waist up, facing right. They are wearing a headset with two earpieces and a microphone. Their right hand is on their hip.

WHITE PAPER

NetPeople[®] Platform Technologies

March 17, 2005

Contents

Introduction	1
Motivation	2
Natural Language Processing	2
Processing	2
Dialogue Act.....	2
Concept List	2
Abstraction with Concepts	3
Reasoning Engine	3
Context.....	4
Goals	4
Operators.....	4
Persistent	5
References.....	6

Introduction

In April of 1991, Bill Gates said:

“AI is helping us create more natural user interfaces . . . we need future software to listen, see, reason and understand the user’s context, intentions and goals.”

Apple says, on their website (February 2004):

“You don’t have to be a scientist to know that the computer of the future will talk, listen and understand.”

The endeavor of computer scientists to improve the human-computer interface is not a new one. Since the earliest days of green-screen dumb terminals, to the fanciest windowed environment and pointing devices, programmers have strived to make the user interface of computers easier and more powerful. Natural language processing, speech recognition and text-to-speech engines have all been pressed into service, severally or together, to try and create the human-like conversations that we know from movies such as “2001: A Space Odyssey” and “Star Trek.”

The sad truth is that, to date, no system can communicate as well as HAL, despite the fact that we’re a few years past his supposed construction date. This issue, human-computer interface, remains one of the most important challenges that face the computer industry. Much of modern computer systems are devoted to the problems associated with information dissemination; from internationalization and portability issues to visual appearance and user input mechanisms, the complexity of these components continues to grow. To date, the part of human-computer interface that is getting a lot of attention from major software developers is natural language interaction. Natural language input has a rocky history in commercial software, but has garnered a lot of research and development over the past few decades. Many experts foresee natural language interface as one of the most important challenges facing the software industry.

Another important aspect for the future of human-computer interface is speech recognition. To date, speech recognition has been used mostly for simple commands (replacing common menus in applications, for instance) and transcription via dictation. Industry leaders, including Microsoft and IBM, believe that speech recognition will be the primary means of controlling a personal computer in the future.

Motivation

The goal of NetPeople is to enable continuing natural language interface between a user and a digital agent, or to put it more conventionally, to enable human-like conversations with an application. This paper will discuss three technologies that help the system make interactions with NetPeople agents more natural seeming.

Natural Language Processing

What separates agent technology from straight-forward media presented information is the ability to accept input from the user and decipher some degree of meaning from it. Within our technology, we refer to information that comes from the user as a *user utterance*.

Processing

Traditionally, agent technologies use a technique for interpreting user input called keyword spotting. This has the advantage of being quick to implement, and relatively easy to understand when creating content. The downside is that it is not easy to distinguish between two similar sentences without a lot of complex pattern matching, and the complexity of the agent's content rises arithmetically with the sophistication of the interaction required.

NetPeople instead uses a more advanced technique: natural language processing. Every utterance is processed to determine both the type of sentence (referred to as the dialogue act¹) and the important concepts that it contains (the concept list).

Dialogue Act

The dialogue act is a way of categorizing input so that it can be matched more accurately against meaning. NetPeople uses two levels of categorization in its representation of dialogue act. The first level simply distinguishes between interrogative (*question*) and declarative (*statement*) sentences. The second level further categorizes questions or statements, such as "information," "desire" or "procedure."

Concept List

The concept list contains all of the ideas likely to be relevant to a sentence. There is a subtle distinction that must be made between words taken from the original user

utterance, and the resulting concept list. NetPeople distinguishes between keywords and the concepts they represent. In its simplest form, this can be merely like a thesaurus, allowing the user to substitute synonyms for words without requiring the content creator to create separate matches for every variation. However, it can also be used in two other ways in NetPeople. The first one is simply a pragmatic way of dealing with the fact that users are rarely perfect in their typing, so common misspellings can be accommodated without forcing the user to be perfect. Some words, such as *separate*, are frequently misspelled, while there are also some very common typographical errors (reversing the 'r' and 'h' in *the*, for instance). The same technique can also be used to handle differences in spelling in different dialects of a language ("litre" for International English versus "liter" for U.S. English, for instance), or of slang. By default, as an example, the English version of NetPeople defines many words that all map to the concept "yes" ("yeah", "yup", "y", etc.), while the Japanese version has several variations of "はい" ("hai").

Abstraction with Concepts

The more sophisticated use of keyword to concept mapping, though, is that of abstraction. A NetPeople concept can represent a group of words with a conceptual connection that goes beyond being synonyms or alternates of each other. A useful illustration of this idea would be to consider the example of a NetPeople agent designed to help users purchase paint. One of the most important functions for this agent is to help the user select a colour for the paint they wish to get. Rather than creating separate optional goals for every potential paint colour, the content creator can create a single concept of COLOUR (as shown at right), and associate all the possible colours with it. This enables content to be written so that it only has to match one concept, rather than a multitude. One content node can be written to match on the concept, so that "I like red" and "I like white" would both match. The action performed in response can still retrieve the actual colour specified if necessary (for storing in a database, for example).

Colour	
Red	Blue
White	Green
Orange	Taupe
Carmine	Eggshell
Aquamarine	Emerald

Reasoning Engine

NetPeople's reasoning engine uses a sophisticated model that makes use of an elaborate state machine and parallel event and goal queues to process agent content very efficiently. Every user interaction (utterance) and every agent response are events

in the system, and intermixed with system events (such as timers, communications from other systems, user written goals, etc.).

Context

One of the most important aspects of state that the system maintains is conversation context. This allows the system to understand ambiguous statements by knowing the current subject of a conversation. For example, if the current conversation is about the Toronto Maple Leafs, the user can ask “what was last night’s score” and NetPeople will be able to properly understand what is meant by that without needing to clarify which score the user is talking about. Much like human conversation, though, NetPeople doesn’t stick to context rigidly, nor will it automatically forget context just because the conversation diverges briefly. As the example at right explains, even though the user didn’t answer the agent’s question right away, NetPeople was still able to associate the answer with the previous question. Naturally, if a real conversation diverges from a topic for long enough, a person would be confused by a sudden answer to a much earlier question. Similarly, NetPeople limits how long it will understand a statement as being associated with a question that hasn’t yet been answered.

Agent: It’s almost time for dinner. Would you like me to recommend a place for you to stop to eat?

User: What time is it?

Agent: Ten to six.

User: Okay, sounds good.

Agent: Great, there’s a good place just three kilometers ahead.

Goals

By using goals, NetPeople allows content creators to create abstractions of real world objectives. There are two forms of goals in NetPeople: operators, which are trivially satisfied, and persistent, which remain until their success criteria are met.

Operators

The former type, called Operators in NetPeople, can be thought of as “instantaneous” tests. If a particular state exists, some behaviour is performed. However, if the Operator’s target state does not exist, it is discarded without further processing. Sometimes, Operators are created with “always true” criteria, to allow behaviours to execute with NetPeople’s normal queued-event model. This allows for other non-interrupting events to be processed before a new behaviour is started.

A good illustration of how an Operator might be used as part of a business process would be a scenario where a NetPeople agent is assisting a user through a purchase

process. In response to the user's expressed desire to "check-out", the agent's content could create a number of Operators representing prerequisite operations for the check-out procedure to complete. These goals, for instance, could check for the absence of shipping information, gift receipt requirements, sales tax calculations, and so on. If any of the goal states matched, the appropriate behaviour for collecting the information would be executed. Note, however, that in this scenario, the goals would have to be re-created at the end of each such behaviour, since by the time the information was collected, all of the Operator-type goals initially created will have been discarded by NetPeople.

Persistent

A higher-level goal is also available in NetPeople: Persistent goals. These allow the content creator to establish a target state that causes a particular behaviour to be executed, in the same way as Operator goals. Unlike the latter, however, Persistent goals (as one would expect) are not discarded until their criteria is met. To re-visit the check-out example given above, it becomes obvious that Persistent goals can be more powerful. This table illustrates how a hierarchy of Persistent goals can abstract the check-out process:

Goal Name	Criteria	Behaviour
SaleComplete	ShippingInfo = Done	
	GiftReceipt = Done	
	SalesTax = Done	ShowFinalSale
GetShippingInfo	ShippingInfo = Empty	CollectShippingInfo
GetGiftReceipt	GiftReceipt = Empty	CollectGiftReceiptInfo
GetSalesTax	SalesTax = Empty	CollectSalesTaxInfo

The SaleComplete goal is not satisfied until the three subordinate goals, GetShippingInfo, GetGiftReceipt and GetSalesTax, are all satisfied. If for some reason, such as a prior check-out by the same customer, one or more of the subordinate goals area already satisfied, the associated behaviours will not be executed.

References

¹ Gustafson-Capková, Sofia: Speech Act Theory in Dialogue Coding, *Department of Linguistics, Computational Linguistics, Stockholm University*