A Distributed Framework for Wide-Scale Domain Independent Natural Language Based Human-Computer Communication

Nicholas Underwood – Georgia Institute of Technology
gte908i@prism.gatech.edu

Abstract
This document proposes a novel framework for a wide-scale natural language communication system between humans and computers. The motivation for the system is two-fold. The first is a desire for a simple system for *low-bandwidth information navigation* (further described in the paper), which would only be one example of the many possible applications that the described system could support. The second being a need to overcome the problems of enormous amounts of knowledge engineering that would be needed on the back-end of a truly domain independent natural language communication system. The framework described confronts this problem by proposing a distributed system much like the web, which would be able to grow over time and would provide a decentralized and flexible mechanism for the knowledge engineering dilemma.

Introduction

A system that would allow natural dialog between man and machine is by no means a new idea. Such systems are common place in science fiction and are on the minds of many computer scientists and non-computer scientists alike. Decades of research have attacked this idea and many of the problems involved with such a system have been more-or-less answered, while many more problems and questions have been raised. It is not clear exactly how far away a system like this might be, though it is certainly clear that much work is left to be done. Current and past research has roots in many different disciplines ranging from human cognition to linguistics to statistics. This paper will layout a new system that would accommodate any and all techniques or approaches to natural language communication, which would almost certainly be a necessity for a system of this scale. This leads to the point that what is being proposed is not a solve-all solution to this problem, nor is it intended to be. Rather, it is best described as a framework to facilitate rapid and distributed growth and development of natural language communication content, research and applications, wherein the end results could be a very powerful tool for communication between humans and computers.

This paper is presented as a broad proposal for a novel system and few concrete implementation details will be discussed. The remainder of this paper will begin with a brief description of the major components of the system, followed by example scenarios intended to further solidify the goals and functionality of the proposed system in the mind of the reader. There will then be some exploration into the requirements such a system would need to be viable and successful, as well as possible limitations or challenges facing the system.

System Components

The system envisioned in this paper consists of three primary components. In the simplest terms, these components could be described as a client layer, a transaction layer and server layer, a familiar architecture for many applications. The client would act as the user's front-end into the system. The transaction layer would handle communications between the clients and the servers. The servers would provide much of the dynamic processing. Furthermore, there could be multiple types of clients and multiple types of servers with the requirement that they interface with a standardized transaction layer. The only other requirement would be some level of network connectivity on the client and server ends (note that in most scenarios this means an Internet connection, though it would not be a necessity for an implementation of the system).

Client Layer
The client layer would take on two roles. The first would involve a mechanism for taking natural language input from the user. This could include speech recognition or be as simple as typed text entry. The second role would be to provide the output from the system, which again could include speech synthesis or be as simple as text printed to a screen. For the most part, the client would do minimal processing of the input, acting mainly as the interface into the system. The majority of its processing would be involved with how to handle the inputs and deciding which server the inputs need to be directed to, based on the input itself. This does present some problems of its own, which will be addressed in further sections.

Transaction Layer
The transaction layer would act as a mechanism for transporting inputs to servers and then returning the outputs from the servers back to the clients. Ideally the interfaces of this layer would be standardized, thus allowing the clients and servers to take on virtually unrestricted shapes and forms.

Server Layer
The server layer would take on the brunt of the dynamic processing involved with natural language processing. Individual servers would accept inputs routed to them by the client layer and then return outputs back to the clients. This is where the power of the system becomes evident. Since the only restraints on the server side is that it interfaces into the transaction layer, servers can be implemented with any techniques or technologies that their developers desire.

Putting It All Together
It still may not be evident why such a system would be useful, so let's talk about the World Wide Web. In essence, the web has the exact same components as described above, only the types of inputs and outputs are different. The client layer typical involves a web browser, the transaction layer includes HTML and HTTPS, and the server layer is typically composed of web servers serving up web pages. Now, if you described the web as only these components it would not seem that interesting or useful. However, everybody knows the web has become one of the most useful systems to ever come out of computing. It's the fact the web is as only as good as the amount and content and diversity of content on the server layer, which these days is enormous, that is important

here. The value of the web grows as the content grows. The exact same would hold true for the system proposed here. As more content, in the form of different domains and types of information involved, grows, so does the value of the system.

Also, if you take a step back and look at the big picture here, another key insight about such a system can be seen. Imagine what it would take to incorporate the vast amount of content available on the web into one single completely contained system. In reality it would most likely be impossible. Now, if you look at one of the most limiting factors at creating a truly domain independent natural language communicating AI system, the amount of knowledge the system would have to entail, you see the exact same problem. The web can be described as a domain independent system because virtually all domains are covered by the content on it. The natural language system describe here would also become domain independent as the amount of content on it grows. The great thing about this is that current research in domain specific natural language systems do very decent jobs at communicating about their respective domains. Add a bunch of these together, and viola, the user is able to use natural language to communicate with the same interface across multiple domains.

Usage Scenarios

Now that the proposed system has been laid out at a very high level, it is possible to present a variety of scenarios which demonstrate its function, usefulness and range of applications.

Scenario 1 – University Information Guide

Background: For students, faculty or even guests of a university there are many different bits of information about the university that are need from time to time. This includes things like office phone numbers, building locations, class times, email addresses, etc. Today this type of information is generally located via the university's web site, which is not a bad way to do it. This scenario is just intended to show how such information location could be done in the proposed system.

Setup: Imagine a new student that has just moved into her dorm for their first semester of college. Tomorrow is her first day of class and she needs to know where and when her classes are. Included in their orientation package was the address of the university's information language server. She opens up her language browser (a text-based client layer similar to the feel of an instant messaging application) and tells it to connect to the server.

Example Dialog (annotated):

```
Student: connect me to gu.edu

(System connects to the server, retrieving certain meta-data about
the server for future reference. i.e. server name, keywords
attributed to the server, initial response, etc.)
```

```
Browser: You are now connected to Generic University Information.
Would like to see a list of top level commands?

Student: yes

Browser: Generic University Information Top Level Commands
Campus Phone Numbers
Building Locations
Class Schedules
Sports
Transportation
Student Activities

Student: i need to look at class schedules

(Server recognizes ‘class schedules’ while being in the top level
state, and moves to ‘class schedules’ state, returning the initial
response from this state.)

Browser:  Would you like to search by class name or class number?

Student: number

Browser: What are the class numbers that you need information on?

Student:  cs101 phys101 eng102

(Server knows that it is expecting class numbers, thus recognizing
the 3 classes.)

Bowser: Class Information

CS101 – Introduction to Computing
Location: That Building
Times: MWF 6:00-7:00 AM
Teacher: That Guy (thatguy@gu.edu)

PHYS101 – Introduction to Physics
Location: This Building
Times: TTh 11:00-12:00 AM
Teacher: Prof. McGuy (mcguy@gu.edu)

ENG102 – English 2 Fundamentals of Verbs
Location: Building 2A3B
Times: TTh 12:00-1:00 PM
Teacher: Some Gal (somegal@gu.edu)
```

This scenario shows a student doing a simple *low-bandwidth* information search. What is meant by this is that very little information is desired by the user, and very little information is necessary from the user to get to this information. This is something that the web handles well, though the web’s level of presentation is not absolutely necessary for such situations. Not going into the details of the server’s implementation, this should simply demonstrate the type of interaction and dialog intended with the proposed system.

Scenario 2 – Weather and a Movie

Background: Weather and movie information are two very common examples of low-bandwidth information. This information can already be easily accessed from a variety of resources including the web. The purpose of this scenario is to demonstrate how the proposed system could handle information across multiple domains.

Setup: The same student in the previous scenario has now been in school for 2 weeks and decides its time to get out of the dorms for a little social activity. She has heard rave reviews about Starship Troopers 3 and would like to go see it tonight, but is a little worried about the weather since she has to walk to the theater. Once again she resorts to her language browser to get the information.

Example Dialog (annotated):

```
Student: i need to know what the weather will be like tonight

(The client recognizes that the term 'weather' matches a keyword of
a previous server that has been accessed before and passes the text
on to that server.  The server recognizes meta-data passed from the
client system and recalls the student's default location.)

Browser:  The weather forecast for Anycity, USA for the evening of
12-12-08.

Temp: 74 - 78 F
Precipitation: 10% chance of rain
Wind: 0 - 5 MPH from North

Student:  no, I need the weather for Genericcity, USA

(The student forgets that her default location is still her home
town. The server knows that in its current state the user might
need weather for other locations, and recognizes Genericcity, USA.)

Browser: The weather forecast for Genericcity, USA for the evening
of 12-12-08.

Temp: 65 - 67 F
Skies: 30% chance of rain
Wind: 10 MPH from East

Would you like to make this your default location from now on?

Student: yes

(The system then updates her meta-data listing for her default
location.)

Student: now I need movie times for Starship Troopers 3

Browser: I'm sorry, I don't understand your request. Would you like
a list of current level commands?
```

```
(The student realizes that she is still connected to the weather
server and forgot to indicate that she wanted to go back to a top
level state in the client.)

Student: top level: movie times for Starship Troopers 3

(The browser recognizes 'top level:' and the keywords 'movie
times', thus returning to a top level state and connecting to the
server attributed to 'movie times.')

Browser:  Please select from the list of local theaters:
1 AMC 56 Downtown
2 Sal's Movie Theater
3 Some Other Movie Place

Student: 1

etc…
```

This scenario shows a plausible way of handling multiple domains and efficient transfer from one domain to the next; a key piece to the success of the proposed system.

Scenario 3 – Another PDA

Background: Personal data management has become a large market within the technology industry. Realizing the potential with the language browser approach to information access and navigation, a new company, Wilbe Rich, Inc., has decided to develop a personal data assistant based on natural language communication. They create a product that easily interfaces with the mass-adopted standardized framework proposed here.

Setup: Our student is now moving along through academia, amassing many contacts and information that will be useful to retain for future use. She hears of Wilbe Rich's new product for personal data management that she can access through her language browser and decides to give it a shot.

Example Dialog (annotated):

```
Student: search for Wilbe Rich

(The client recognizes the top level command 'search' and locates
Wilbe Rich's server.)

Browser: The Wilbe Rich Personal Data Assistant has been located.
Would you like to connect now?

Student: yup

Browser:  Welcome to the Wilbe Rich Personal Data Assistant, do you
have an account with us?

Student: no
```

Browser: Ok, would you like for us to create an account for you?

Student: yes please

(The browser uses the meta-data passed to about the student to create a new account.)

Browser: We have created a new account for you. Your login will be your email address. Please enter a password now:

Student: herbert

Browser: Please repeat the password now:

Student: herbert

Browser: Thank you, your account is now ready. Would you like a tour of the features available to you?

Student: no thanks

Student: top level commands please

Browser: Personal Data Assistant Top Level Commands
Enter new contact.
Search contacts.
Add event to schedule.
Browse schedule.
Search schedule.
Transfer money to Swiss bank account.

Student: help me enter a new event

(Server recognizes 'help' and 'new event and switches to the help with new contact state.)

Browser: You can enter a new contact by saying 'Add new event <eventname>' The system will then prompt your for more information about the event.

Student: Add new event English Class

Browser: Is this a ONETIME, WEEKLY, MONTHLY or YEARLY event?

Student: weekly

Browser: What days of the week?

etc…

(Later on)

Student: what is the weather like?

Browser: ALERT: You have English class in 5 minutes.

```
The weather forecast for Genericcity, USA for the evening of
12-12-08.

Temp: 65 - 67 F
Skies: 30% chance of rain
Wind: 10 MPH from East
```

This scenario shows how the system could be used for functions other than information search and navigation. The alert is also intended to demonstrate that system could also be proactive, not just reactive.

Scenario Conclusions
The above scenarios cover only a small subset of the types of applications that could be applied to the framework. It is the inherent flexibility of the framework that gives it its power. The above scenarios also only cover applications that are already available and proven on the web, and indeed the majority of the applications that the framework could support are and would be able to be handled by way of the web. However, there are a couple of key advantages this system would have over the web as described in the following section.

Language Browsing vs. Web Browsing
Feedback-based Navigation: This framework provides a stronger sense of feedback-based information navigation via dialog, while the web framework puts the onus of navigation on the user and site designer. While it would still be up to the quality of the dialog design in the proposed framework, the ability of the system to ask the user questions for clarification can greatly increase the efficiency of this process.

Extraneous Noise: The web can provide users with endless amounts of low-bandwidth information; however, often included with this information is extraneous noise in the form of images for presentation, advertisements, links to related information, etc. While not always the case, most of this noise is unnecessary to satisfy the user's information needs and sometimes a hindrance or an annoyance, slowing down the access to the desired information. It's not to say that web information sources couldn't be architected to avoid this noise, it's just that that's not generally the case. This framework could eliminate much of this noise simply because it doesn't concern itself with presentation, only the pure information.

Cognition/Learning Curve: Since this framework is based on pure language and dialog for information access and navigation, a process already familiar to humans, it is reasonable to assume that it would provide a more natural and intuitive interaction across all information sources. When using the web, users are forced to interact primarily with clicks only and must learn the site structure for each information source for information navigation.

Requirements for a Successful System
The three key benefits of a natural language based system over the traditional web approach described above provide some sense of why such a system might be useful, however, there are a set of requirements the system would have to satisfy to make it

successful. The following three sections, separated into the separate system layers, layout some of these requirements.

Client Layer Requirements
Top Level State: The client must be able to recognize a basic set of top level commands or keywords when control of the dialog is not passed to a particular server. These commands/keywords should be standardized and should allow the user to easily and naturally connect to the appropriate servers.

Meta-data: It is important that clients have the ability to maintain certain pieces of information about the user and the system. This information will make it easier to provide more seamless transfer from one domain/server to the next. It will also provide the system with a sense of a memory and personalization, functioning much like cookies for web systems.

Keyword Management: One of the key failures in the design of the web is that searching for information is hard. If keywords or something similar had been built into the web architecture, search would be an easier problem. For the framework described here, keyword management is a necessity for more *natural* communication. While a user is in a top level state in their client, the client must be able to interpret which server to send their commands to. A simple approach to this is to have keywords that are assigned to servers, and when the keyword is recognized, control of the dialog is passed to the assigned server. Managing these keywords efficiently and keeping the users in control of the keyword-server requirements is necessary.

Searching for New Domains/Servers: Built-in at the top level state should be a mechanism for searching for information on domains/servers not yet seen by the client. This is a crucial piece of the system to ensure usability, and would likely be one of the most challenging aspects of the system to get right.

Transaction Layer Requirements
Standardized Interfaces: The most important aspect of the transaction layer would be standardization. This is vital for allowing a range of different clients and server types.

Unrestricted Meta-data: The transaction layer should be able to handle any and all types of meta-data between the client and server layers, leaving it up to those layers to interpret the meta-data as needed. This will ensure the flexibility necessary for a transaction layer.

Server Layer Requirements
Dialog Control: Managing conversation and dialog is key to a usable server layer. The example scenarios assumed internal state transitions, with ELIZA-like language rules for each state. This is only one possible approach, and the flexibility of the framework would accommodate any approach.

Error Handling: Since servers are only expected to be able to communicate about a single domain, it is important that they have built-in mechanism for handling un-related or senseless inputs from the client.

Challenges

There are a variety of challenges that would face the wide-scale adoption of this framework. Generating initial interest would top this list. At early stages the limited amount of content, the driving force behind its true value, would make it appealing only to a select few. Much like the internet and the web, it would likely have to start in academia and grow from there. A second, and possibly more crippling, challenge would be the lack of business models that could be applied to the framework to support commerce, primarily advertising based commerce. Advertisements could be injected into the dialog, and likely would be, however this would be the extra noise that the framework inherently tries to avoid. Shopping applications and fee-based services would have to drive commerce in the framework, but mass adoption would have to be a precursor for this. In early stages, open, wikipedia-style resources would have to drive growth.

Conclusion and Potential

The framework described herein provides an alternative way of looking at natural language based human-computer interaction. Current research tends to focus on single-technique cure-all methods for handling truly domain-less human-computer communication. The major take-away point of this paper and this framework is that there is probably a much better way to approach this problem, and now with the ubiquity of the internet, that way may be through distributed and flexible techniques as described here, even if it doesn't take the exact form as this paper describes. Much work would have to be done to get something like this off the ground, but if it were to manifest it would have endless potential.

As a final thought, it should be noted that as described in this paper, only the surface has been scratched when it comes to the true potential of a system like this. Throughout the paper, the system has primarily been discussed as a personal computer application. However, the true colors of this system only show when looking past the personal computer interfaces. The beauty of such a system is that since it is based on natural language for interaction, a typical computer interface is completely unnecessary. Attach speech recognition and text-to-speech to a client and the system can be accessed from virtually anywhere at anytime requiring only an internet connection. Place a client on a cell-phone, attach a client to OnStar like systems, or even attach a client to a 411 like phone number and immediately natural language access to limitless information becomes completely ubiquitous.