A Statistical Approach to Multimodal Natural Language Interaction

John Vergo

IBM Corporation/T.J. Watson Research Center
30 Saw Mill River Road
Hawthorne, New York 10532
(914) 784-7035
jvergo@us.ibm.com

Abstract

The Human-Centric Word Processor is a research prototype that allows users to create, edit and manage documents. Users can use real-time continuous speech recognition to dictate the contents of a document. Speech recognition is coupled with pen or mouse based input to facilitate all aspects of the command and control of the application. The system is multimodal, allowing the user to point and speak simultaneously. In particular, the correction, formatting, organization and manipulation of dictated text are greatly facilitated by the combination of natural language understanding and multimodal input. The system uses a maximum entropy, statistical approach for mapping a combination of natural language and pointing events into multimodal formal language statements.

Keywords

Multimodal input, natural language processing, speech recognition, gesturing.

Introduction

The Human-Centric Word Processor (HCWP) is work that grew out of the MedSpeak/Radiology project (Lai and Vergo 1997). The MedSpeak system was the first commercial real-time continuous speech recognition system. Radiologists use the system to dictate radiology reports. Speech is used to dictate the reports and to navigate through the application in a "hands free" and "eyes free" manor.

Usability field studies of the MedSpeak/Radiology system identified two important areas for potential improvement. MedSpeak uses a very constrained dynamic vocabulary for command and control of the system. Although simplicity was a primary design goal, it was found that users (radiologists) had significant problems remembering the specific commands that were required to navigate the system.

The second area for potential improvement is in the correction of speech recognition errors. Working in a "hands free" environment means radiologists do not use a keyboard to do their work. MedSpeak has a 96% accuracy rate for dictation, which means that for a typical 300 word report the radiologist has to correct 12 errors on average. This requires significant manipulation of the keyboard and mouse.

Radiologists are an unusual user group, in that they are experts at dictation. They typically dictate 100 to 150 reports a day. Because of the large number of reports, and because the reports are relatively short, they can mentally compose well-organized text in their minds and dictate the report cleanly to the MedSpeak application.

When considering dictation applications for the general public, the dynamics change considerably. Document sizes may increase dramatically, and the vast majority of the people are not trained on how to dictate. The result is that much the text produced by dictation requires considerable manipulation after the dictation has been completed. The manipulation might include rewording sections of text, formatting changes, and significant organizational changes. With commercially available systems today, making these types of changes requires extensive keyboard manipulation and/or cumbersome and error prone navigation through the document by voice.

The HCWP strives to address the issue of radiologists having to remember specific commands by employing natural language understanding (NLU). The NLU engine (Papineni, Roukos and Ward, 1997) developed at IBM research, uses a maximum entropy approach to the translation of unconstrained natural language statements to application specific formal language statements. This process is described in greater detail below.

The HCWP addresses the issues of error correction and post-dictation manipulation of text by allowing actions to be constructed multimodally, using speech and stylus.

Typical multimodal user interactions

These are some typical examples of multimodal constructions that are handled by the HCWP:

Example 1.1 "Delete this word"
Example 1.2 "Underline from here to there"
Example 1.3 "Playback this paragraph"
Example 1.4 "Change this date to the third"

Examples 1.1, 1.3 and 1.4 are all accompanied by a single pointing event. Example 1.2 is accompanied by two pointing events. In Example 1.3, the user is requesting the system to play back the audio associated with the indicated paragraph. In Example 1.4, the user need only point in the vicinity of a date for the system to understand her. In all cases, the ability to express these requests multimodally results in greatly reduced task completion times when compared with traditional GUI techniques or speech only systems.

Architecture

A taxonomy of multimodal integration mechanisms has been presented by (Blattner and Glinert 1996). They describe three general approaches; frame integration, neural nets and the use of agents. The HCWP combines some aspects of all three approaches in handling multiple input modalities. Figure 1 gives an overview of the functional aspects of the system.

The NLU engine has neural net-like features in that it operates in two distinct phases. First, it must be trained using a corpus of training data. It is then used to translate new natural language statements into formal language statements.

The formal language produced by the NLU engine is similar to the concept of a frame, in that formal language statements contain objects, each of which have attributes. This will be illustrated in greater detail in the sections on natural language understanding and augmented formal language.

Finally the HCWP directly implements agents to process actions that are represented by the formal language.

Speech Recognition

The HCWP system uses the IBM ViaVoice continuous speech engine, along with an object oriented framework that completely encapsulates the speech engine functionality (Srinivasan, Vergo 1998). The HCWP system places the engine in one of two modes. In dictation mode, the speech engine is given a specific vocabulary that it uses to decode speech to text. The decoded text goes through a formatting process, and is sent to a multi-line editor (MLE) control, where it is

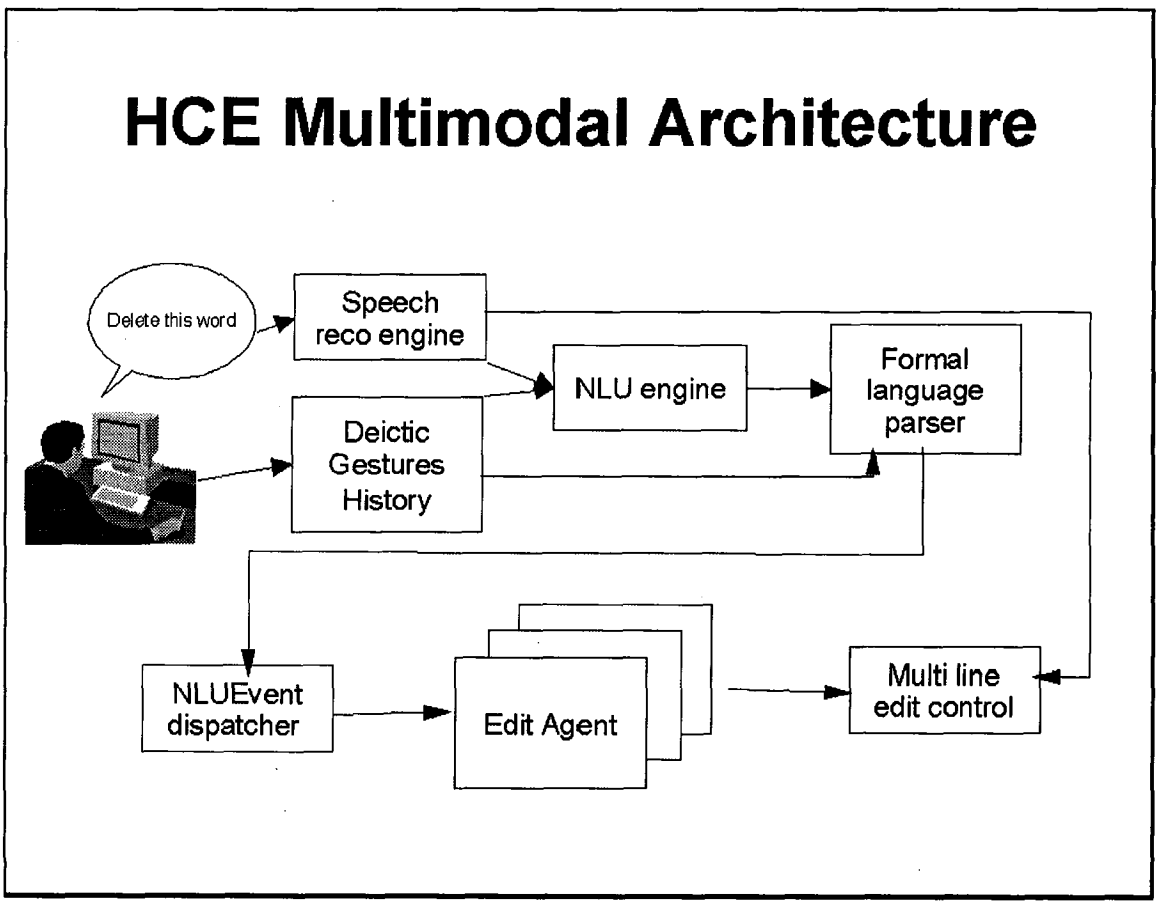

Figure 1

displayed to the end user. Formatting typically includes capitalization, putting the correct spacing between words, punctuation, etc.

The second mode is navigation mode, where the user is controlling the application. This is the mode that we concern ourselves with for the remainder of the paper. In this mode, the spoken commands are not displayed in the MLE, but are ultimately turned into actions that are performed by the system. In navigation mode, the speech engine is again decoding text from a dictation vocabulary, but the vocabulary is especially built for the navigation task. It is a large vocabulary (>64,000 words), approximately the same size as the vocabulary used in dictation mode. With this large recognition vocabulary the user can phrase requests for actions to the system in an unconstrained manner. This is in stark contrast to a dynamic vocabulary or grammar based approach to speech recognition.

With dynamic vocabularies, the user speaks single words to affect actions in the application. For example, to traverse a menu, the user might have to say "File", "Print" and "OK", each quoted word being a single word in the dynamic vocabulary. Alternatively, "Print" might bring up the print dialog directly, without having to traverse a GUI menu structure.

A grammar based approach allows the users to speak in predefined constrained phrases. For example, the user might be allowed to say "Print the document" or "Print it". These grammatical constructions must be explicitly allowed by the grammar that has been enabled on the speech recognition engine. If instead, the user says "Send it to the printer", and the construct is not part of the grammar, it will not be properly recognized.

Gesturing

The HCWP system currently uses a liquid crystal display (LCD) tablet with a stylus for input. At present, the only gesturing supported by the system is deictic gesturing (i.e., pointing events with no semantic content contained therein). Using a stylus to accomplish spatially oriented tasks has been shown to be a natural mode of interaction (Oviatt 1997), with 100% of the tested user group having a preference to interact multimodally during a map task.

Deictic gestures are detected as asynchronous events and are stored in the deictic gesture history (Figure 1). The gestures are stored with context dependent information. For example, if a user points at the MLE, character position and selection information are saved along with the event. If the user gestures at a list of files, indices are stored along with the event.

Natural Language Understanding

The natural language understanding engine uses a feature-based language understanding approach [Papineni, Roukos and Ward 1996]. The engine uses a statistical translation approach that translates a natural language statement into a formal language statement. Formal language statements are well-defined statements that the application can accept as an "action" to perform.

In the development of the prototype, we collected 700 English requests that users made while interacting with a low fidelity prototype of our system. The sessions were video taped, and each statement made by the user was transcribed and then mapped (by hand) to a formal language statement. The natural language statements and their formal language mappings constitute the training corpus. Two examples:

N1 Get rid of the last sentence
F1 [delete] target(locator(sentence last))

N2 Change entertainment to education
F2 [change] target(locator(string "entertainment"))
newValue(locator(string "education"))

If n is a natural language statement, f is a formal language statement, and F is the space of all formal language statements, then the NLU engine maximizes the conditional probability $P(f|n)\ \forall\ f \in F$.

The training corpus is used to build models whose parameters are "trained" automatically from the training data. Given a sufficiently large training corpus, statistical translation models are built which are capable of translating new sentences in the natural language to formal language statements with high accuracy.

Augmented Natural Language

The interpretation of *multimodal input* begins with the augmentation of natural language with gesturing information. The gesturing information accompanies the natural language as an input to the NLU engine. In this manner, it simply becomes another "feature" for the statistical translator to take into account when evaluating conditional probabilities of formal language statements. Two examples:

Example 2.1

N1 (pointing 1) Delete this word
F1 [delete] target(locator(pointed 2))

Example 2.2

N2 (pointing 2) Remove all text from here to there
F2 [delete]target(locator(start(pointed 5)end(pointed 7)))

Both natural language statements (N1 and N2) have been augmented with a "(pointing = n)" construct. This construct is an indication to the NLU engine of how many pointing events were associated with the natural language statement. The NLU engine responds with a formal language statement that contains formal language constructs for deictic gestures. For Example 2.1 formal language statement F1 contains the construct "pointed 2". This *formal language* construct may be read as "the pointing event was associated with the second word". The construct is then used to align pointing events (stored in a deictic history buffer) with sub-elements of formal language statements. We can do this because we have timing information associated with each spoken word and pointing event.

Example 2.2 provides further illustration. N2 includes the construct "pointing 2", giving the statistical translation engine a strong hint that the statement was accompanied by two gestures. The engine produces a formal language statement with a "locator" component that has two "pointed n" constructs. The first, "pointed 5" can be interpreted as meaning the start location for the deletion is indicated by the pointing event that took place when word number 5 of the natural language utterance was spoken. Similarly, the second deictic component "pointed 7" can be interpreted as meaning the end location for the deletion is indicated by the pointing event that took place when word number 7 of the natural language utterance was spoken. Since all spoken words are time-stamped, we retrieve the time information, align with pointing events stored in a deictic gesture history buffer, and calculate the range of characters to delete based on the stored pointing events.

Formal Language Parsing and Natural Language Event Dispatching

The formal language statement produced by the NLU engine is in a flat text representation. The formal language parser accepts the string representation, and transforms it to a heterogeneous collection of objects that are contained in a top level object called an NLUEvent. The NLUEvent is a class that is designed to reflect, in a very direct way, the structure of the formal language. As can be seen from the sample formal language statements above, the formal language structure and content are driven from the tasks that the HCWP is designed to carry out, i.e., it is very domain/task dependent. By necessity, the NLUEvent also contains domain specific features.

After the formal language parser builds the NLUEvent, the event is sent to one of a series of agents in the application that are capable of carrying out the action. The decision as to which agent receives the NLUEvent is made by the natural language event dispatcher. The dispatcher makes the decision on where an event is routed based on a series of criteria.

Some events are "hard wired" to specific agents. For example, printing the document is always sent to the same agent, regardless of the context of the application. Other events can be ambiguous, and require an understanding of the state of the application to interpret correctly. For example, if the user says "Delete this", and gestures while uttering the request, there may be two very different interpretations possible, depending on what he pointed to. The formal language statement

[delete] target(locator(pointed 2))

is the same regardless of the interpretation, as is the resulting NLUEvent.

If the object he pointed to is a word in the body of the multi-line editor, then it is appropriate for the dispatcher to route the NLUEvent to the edit agent. The edit agent only knows how to interpret the NLUEvent in the context of editing actions, and carries out the action appropriately (in this case deleting the word that was pointed to).

The second possibility is the user pointed to a file in a list of files, and said "delete this". Under this circumstance, the dispatcher sends the NLUEvent to the filer agent, which knows how to carry out the requested action in its own context, deleting the file.

The dispatcher makes the decision on where to send an NLUEvent based on the event itself, the context of the dialog between the user and the system, and on the GUI input focus of the application.

Conclusions

At this time, our prototype system is up and running. The 700 sentences we have collected thus far have yielded a functional system with low end-to-end accuracy. This is an anticipated result. The current version of the system will be used to collect significantly more data (minimally, 10,000 sentences are anticipated). Our expectation is to achieve and end-to-end accuracy rate of 80%.

Future work

As a prototype, the Human-Centric Word Processor is in its infancy and provides an excellent platform to pursue research on a variety of topics.

The formal language, the object oriented design of NLUEvents and the agents are all application specific. A challenging area of research is to come up with a design and/or framework where domain independent tools and components can be used and extended into specific areas, facilitating the creation of similar types of applications. The augmentation of natural language and the formal language must also be expanded to include a much wider

array of pen based input gestures (i.e., going beyond simple deictics). Other input modalities need to be explored, including vision, eye tracking, etc.

The architecture of the HCWP assumes that speech is the "triggering" modality. Clearly this is not always the case. Users may completely specify actions using pen or vision based gestures. The architecture needs to reflect this reality.

Finally, the effect of the size of the training corpus on the accuracy of the system will be of tremendous interest, since the collection of training data is a time consuming and labor intensive task. Data need to be collected and published on this topic.

References

M. M. Blattner, E. P. Glinert, Multimodal Integration. IEEE Multimedia, 3(4), 14-25, 1996.

Lai, J. and Vergo, J. 1997. MedSpeak: Report Creation with Continuous Speech Recognition. In Proceedings of ACM SIGCHI 1997, Conference on Human Factors in Computing Systems, 431-438. Atlanta, Georgia.

Oviatt, S., DeAngeli, A., Kuhn, K., 1997. Integration and Synchronization of Input Modes during Multimodal Human-Computer Interaction. In Proceedings of ACM SIGCHI 1997, Conference on Human Factors in Computing Systems, 415-422. Atlanta, Georgia.

Papineni, K. A., Roukos, S., and Ward, R.T. 1997, Feature-Based Language Understanding. In Proceedings of the 5th European Conference On Speech Communication and Technology Volume 3, 1435-1438. Rhodes, Greece: European Speech Communication Association.

Srinivasan, S. and Vergo, J. 1998. Object Oriented Reuse: Experience in Developing a Framework for Speech Recognition Applications, ICSE 1998, Joint ACM and IEEE International Conference on Software Engineering, Kyoto Japan, Forthcoming