CONVERSE: a Conversational Companion

B. Batacharia, D. Levy, R. Catizone, A. Krotov and Y. Wilks

1 Introduction

Empirical and theoretical investigations of the nature and structure of human dialogue have been a topic of research in artificial intelligence and the more human areas of linguistics for decades: there has been much interesting work but no definitive or uncontroversial findings. the best performance overall has probably been Colby's PARRY (Colby,1973) since its release on the (then ARPA) net around 1973. It was robust, never broke down, always had something to say and, because it was intended to model paranoid behaviour, its zanier misunderstandings could always be taken as further evidence of mental disturbance rather than the processing failures they were.

Colby actually carried out a version of the Turing test (Turing,1950) by getting psychiatrists to compare blind PARRY utterances with those of real paranoids and they were unable to distinguish them—indistinguishability results are never statistically watertight, but it was, nonetheless, a very striking demonstration and much longer ago than many now realise.

CONVERSE was intended not to be based on any scientific research but on hunches about how to do it, together with taking advantage of some recent methodological shifts in computational linguistics. The main hunch was derived directly from PARRY's impressiveness when compared with its passive contemporaries like ELIZA (Weizenbaum,1976): PARRY had something to say, just as people do, and did not simply react to what you said to it. It could be said to rest on the hunch that a sufficient condition for humanness in conversation may be what Searle (Searle,1983) calls intentionality: the apparent desire to act and affect surroundings through the conversation, which is a strong version of what we am calling "having something to say" since a computer program without prostheses can only create such effects through speech acts and not real acts on physical objects.

The extension of this hunch as far as Turing test—i.e. fooling people that the system is human—situations is concerned is that if the computer can get to say enough, to keep control of the conversation, as it were, through being interesting or demanding enough that the human plays along, then there is correspondingly less opportunity for the human interlocutor to ask questions or get responses to an unconstrained range of utterances that will show up the system for what it is. Naturally enough, this hunch/heuristic, must be tempered in practice since a system that will not listen at all, which will not be diverted from its script no matter what is said, is again inevitably shown up. The hunch/heuristic is simply a tendency—be as active and controlling in the conversation as possible, subject to necessary and unavoidable interruptions and topic shifts driven by the human. RACTER, PARRY's only real rival over the last 30 years, worked on the principle of being so interesting and zany that many humans did not want to interrupt it so as to intrude new topics or demands of their own. Others were less charmed of course, but it was one very effective strategy for operating this key hunch and one not involving actual clinical madness.

The original features of CONVERSE are probably as follows:

1. top down control of conversation by means of a range of scripts, plus an active bottom up module seeking to answer questions etc. against a set of data bases on individuals.
2. control and interaction of these features in 1 by means of a weighting system between modules that could be set so as to increase the likelihood of one or other of these modules "gaining control" at any given point in the conversation.
3. the use of large scale linguistic data bases such as thesaurus nets giving conceptual connectivity—for dealing with synonyms---and large proper name inventories that allowed CONVERSE to appear to know about a large range of people and things not in either the scripts, the data bases, or the semantic nets, though this information as formally mapped to the structures of the semantic network and the databases.
4. a commercial and very impressive text parser, based on trained corpus statistics. This however had been trained for prose rather than dialogue which meant that much of its output had to be modified by us before being used. We also made use of large scale patterns of dialogue use derived from an extensive corpus of British dialogue that was recently made available.

The last takes advantage of recent trends in natural language processing: the use of very large resources in language processing and intermediate results obtained from such resources, like the dialogue patterns. It meant that CONVERSE was actually far larger than any previous Loebner entry, and that much of our effort had gone into making such resources rapidly available in a PC environment. So, although not based on specific research, CONVERSE was making far more use of the tools and methods of current language processing research than most such systems. Its slogan at this level was “big data, small program” which is much more the current trend in language processing and artificial intelligence generally than the opposite slogan which had ruled for decades and seen all such simulations as forms of complex reasoning, rather than the assembly of a vast array of cases and data.

CONVERSE, although, it has some of the spirit of PARRY, does in fact have data bases and learns and stores facts, which PARRY never did, and will allow us in the future to expand its explicit reasoning capacity. The weighting system would in principle allow great flexibility in the system and could be trained, as connectionist and neural network systems are trained, to give the best value of the weightings in terms of actual performance. We will continue to investigate this, and whether weightings in fact provide a good model of conversation—as opposed to purely deterministic systems that, say, always answer a question when it is posed. In the end, as so often, this may turn out to be a question of the application desired: a computer companion might be more appropriate weighted, since we seem to like our companions, spouses and pets to be a little unpredictable, even fractious.

On the other hand, a computer model functioning as a counsellor or advisor in a heath care situation, advising on the risks of a certain operation or test, might well be more deterministic, always answering a question and always telling all it knew about a subject when asked.

2 The CONVERSE Personality and Characterization

The character of CONVERSE is Catherine, a 26 year-old female editor for Vanity Fair, who was born in the UK, but currently lives in New York. The details of Catherine’s character are stored in a database, known as the Person DataBase (PDB). The kinds of things that we store about Catherine are the details of her physical appearance, her birthday, astrological sign, some of her likes and dislikes, whether she has a boyfriend, where she works, etc. For the most part, things in the PDB are all related to facts about Catherine. We can also store information about other people in the PDB, in particular people that are related to Catherine in some way, mother, father, friend, boss, etc.

The scripts are the driving force of the program and whenever possible, we aim to keep control of the conversation, by posing a question at the end of a system utterance. The scripts cover a range of 80 topics, but can be extended (within the limits of the hardware). Currently, some of the topics covered are crime, racism, religion, ‘The Simpsons’, mobile phones, abortion, travel, food and violence. The method for acquiring the scripts is done in a two stage process. First, a script writer sketches out the script on paper and secondly, the scripts are entered into the system via a script editor. The script editor establishes the flow of control through the script based on the user’s responses to each script utterance.

3 System Modules

3.1 Overview

CONVERSE consists of a number of independent software modules which are co-ordinated by a shell structure. The total size of the code is approximately 600 kbytes, plus 28.8 mbytes of data. The software can currently converse on a variety of 60 different topics, using the knowledge of several thousand people and places. In addition it can provide answers to questions using the same knowledge base. CONVERSE is implemented in C and C++ and runs under Windows95. It employs a parser written by Prospero Software, a UK company, and a lexical database called “WordNet” created at Princeton University under the direction of George Miller (Miller, 1990).

Very briefly, the system works as follows (see Figure 1). First, user input is read and passed through the input modules. Then, a dispatcher calls up action modules to see if they can do anything with the processed input. Then, it collects the text generated by the action modules, chooses the most appropriate and passes it through the generator, which prints out the output message. Meanwhile, action modules read and update the data in data modules. When the output has been printed, the system waits for the user input and the entire cycle starts again. The remaining part of this section will describe in detail how each of the components operates.

3.2 Architecture

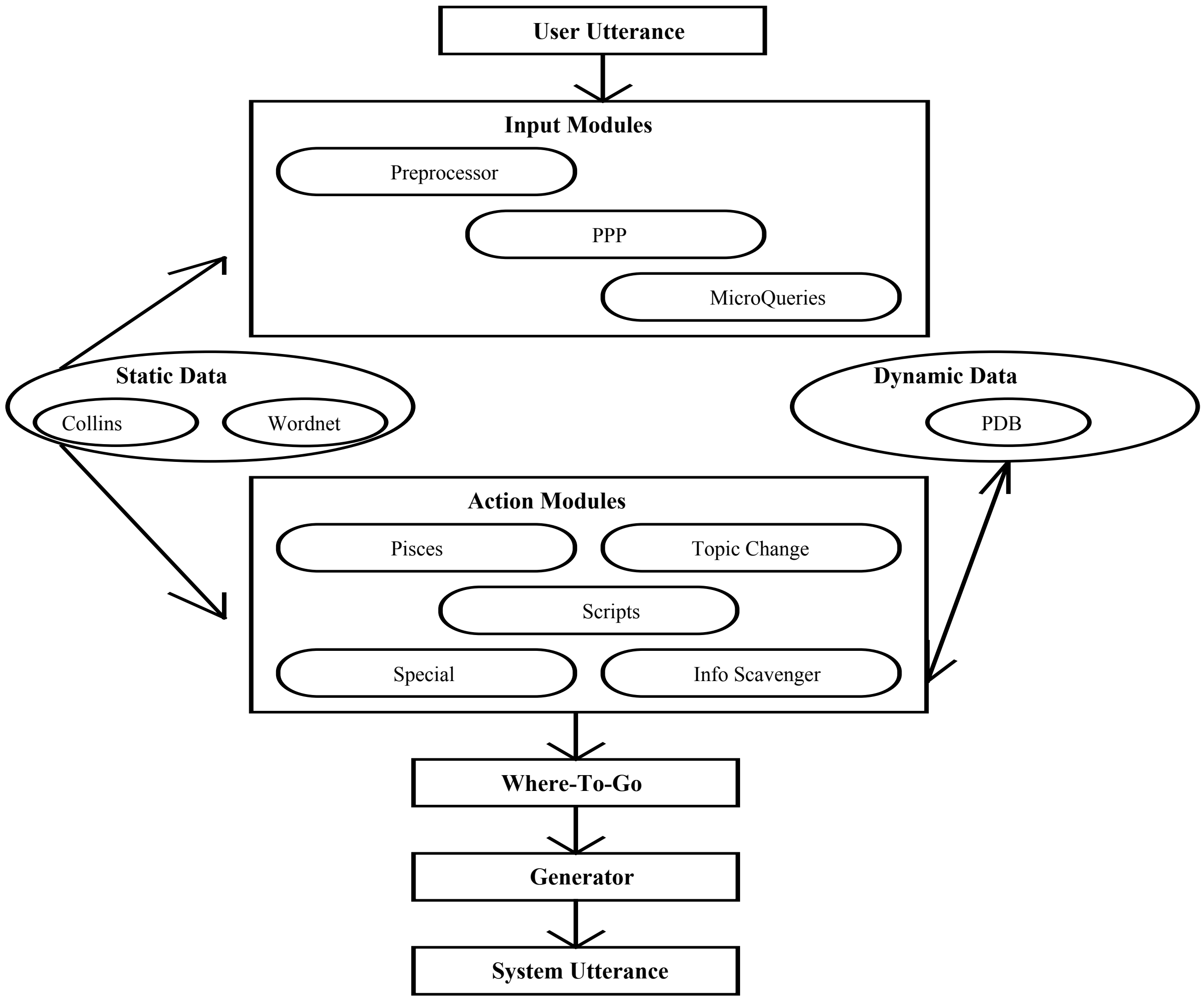

Figure 1

3.3 Input Modules

Input modules deal with the user input. Their function is to extract any useful information from the input and present this information to action modules.

First of all, the utterance is pre-processed in order to correct grammatic imperfections, spelling mistakes and to identify proper and place names. Then, the corrected utterance is passed to Prospero parser and to the parser post-processor (PPP). Finally, the microquery module provides an interface which other action modules use to access the output of PPP.

3.3.1 Pre-processing

Pre-processing includes the following modules: spell checker, elision expansion module, name tagger, and trivial punctuation correction module. Using a large dictionary, spell checker corrects the user's spelling. Elision expansion replaces abbreviations such as 'I'll', 'they've', 'it's' with their full forms (I will, they have, it is). If the input contains any place names or proper names, they are identified by the name tagger. First names are augmented with a sex (or as being unisex). Place names are augmented with extra place information data. In a similar manner the tagger marks dates and arithmetic expressions. Name, place, date, and arithmetic information is used by action modules through the microquery module. For example, if 'John' was typed in response to 'What is your name?', we can tell that the user is probably a man called John. Similarly, we can look for the user's birthdate and birthplace and say something clever about the latter.

Finally, trivial punctuation correction module rectifies a problem where the user input is not terminated with a punctuation sign. The module decides whether the input is a question or a statement and puts the appropriate punctuation sign at the end.

3.3.2 Prospero and PPP

Prospero Software's syntactic parser is used to determine the sentence structure. It gets the pre-processed input, tags it with the part-of-speech information, and then parses. The output of the parser is passed to the Parser Post-Processor (PPP).

The PPP's tasks are two-fold: it simplifies and flattens the parse tree and also corrects the deficiencies of Prospero. For example, given a sentence 'My name is Mary', PPP can tell that the verb of this sentence is "be" and the object is "Mary". PPP also knows that 'Mary' is a female name (from the information supplied by the pre-processor). PPP breaks the input into sentences and determines whether each sentence is a command, question or a statement.

3.3.3 MicroQueries

Microquery module is a front end to the PPP. Each action module queries the processed user input using the list of microqueries. A microquery defines a set of patterns posed as 'questions' to the input in order to filter and extract relevant information. For example, one pattern can determine if there are any proper names in the input. Individual patterns are connected to each other with logical functions. They can access the elements of input as well as Wordnet functions (synonym sets and hierarchies) and built-in functions such as string matching.

Action modules can decide what to do according to what answers the microQueries returned. Microqueries can also input and return values and use variables. For example, a microquery can determine a command "Tell me X" and pass the value of X to the action module for further processing.

3.4 Data Modules

3.4.1 Person DataBase (PDB)

This is the module which is responsible for storing all the information for each user, as well as information about the system's own persona - Catherine.

Pisces answers most questions by fuzzy matching with the Person DataBase. A potential tagset will be formed from the user's utterance. This tagset will be passed to the fuzzy-matching portion of the PDB, which will attempt to find a "closest match" tagset within the database. Given a good enough probability that the tagset matches, Pisces will then be able to supply a response, hopefully answering the question.

This is the module that will be responsible for storing all information for each judge.

Information will be stored with associated tags - it will not be arranged as a slot-filled table. Information that can be stored may be numerical (in either integer or floating-point format) or a text-string. Tags will be passed as text strings. The order of fields is unimportant. There is, also a mechanism for allowing lists. There are procedures to allow for the entire set of the current user's data to be exchanged for another user's data. This is useful in a multiple user environment.

3.4.2 Wordnet

Wordnet is a lexical database created in Princeton under the guidance of Professor George Miller. It stores words and their definitions as well as synonym sets (complete, finish) and hierarchical information (e.g. wine is a kind of a drink). Wordnet is used by PPP, MicroQuery module (for synonym and hierarchy functions) and Pisces (for word definitions).

3.4.3 Collins

We are using the Harper-Collins dictionary of proper nouns for identifying famous people and places. This resource is accessed primarily by the question-answering module, Pisces.

3.4.4 MicroQuery providers

Each action module and each script maintains its list of microqueries. Individual microqueries are stored as text, and are different from the MicroQuery module which applies these microqueries to user input. For example, Pisces (the question answering module) has microqueries filtering out only questions and determining the type of a question. Special case module has microqueries looking for rude words. Some script lines have microqueries extracting sentences specific to that script line.

3.5 Action Modules

3.5.1 Where-To-Go Module

The "Where-To-Go" (WTG), or dispatcher module decides which of the action modules handle the input utterance. The analogy that is the most powerful is that of dispatcher acting as an auctioneer, holding each piece of input up for bids by each of the action modules.

The Dispatcher presents input to each action module in turn, keeping track of the bids. An action module returns a bid describing that module's confidence that it has a valid response to the current piece of input. This bid lies between 0 (no bid—I can't say anything about this) and 100 (I understood completely what the user said and I can respond to it). A piece of input is then 'won' by the module which returned the highest bid. In some cases, there can be more than one winner.

The calculation of a bid depends on each individual action module. They are described in detail in later sections.

After the bidding is complete, dispatcher decides which parts of input will be presented to which action modules. Some parts of input may not be processed at all, in which case the information scavenger module is activated. If an action module wins a bid, then it may extract data from the input and store it in the Person Database.

3.5.2 Pisces

Pisces is the module that answers questions. Pisces functionality is broadly divided into two tasks: question recognition and question answering. The question recognition module tries to identify the type of question asked based on a typology of about 20 different question types. Once the question type is identified, a response to the question is given. The response is formed by looking first in the Person DataBase (using a synonym matching algorithm with Wordnet) to see if we have an accurate response to the question. If this fails, we produce a filler that corresponds to the users question type. If we fail to recognize the question type, a general filler is given. Pisces also handles user requests. All of the request/question type identification is done using microQueries. Both Wordnet and Collins are referenced by Pisces for definitions of people, places and things.

3.5.3 Special Cases

This module handles exceptional cases such as rude language, illogical birthday or age, talk about sex, violence. Once a special cases situation arises in the program, we trigger a response taken from a canned set of phrases that correspond to the situation at hand.

3.5.4 Scripts

Script module guides the conversation through a network of script lines. It keeps track of the “script pointer” pointing to the current script line. Each script line has some canned text, branches pointing to other script lines, and a set of Microqueries together with other information. When the conversation reaches a certain script line, the canned text is passed to the generator, and the system awaits user response. User input is then run through the set of Microqueries to determine how relevant input was to the script line, which branch to follow next, and what information should be extracted from the person database. A bid is then passed to the dispatcher.

Script module keeps track of what lines have been visited by the user. In case conversation comes back to the same line, the person database should have the previous user’s response to the script question. If the response is known, the relevant branch is simply followed. Also, if user input does not match the current script line, another matching line can be found.

3.5.5 Information Scavenger

The function of this module is to extract any possible information from the given clause if the clause is a statement. This module is the penultimate one called by the Dispatcher. It is called only if no other action modules have extracted information from the current clause.

3.5.6 Topic Change Module

Topic Change module controls the flow of conversation. It decides in what order the available 60 topics should be presented to the user. Catherine starts with the introduction, and then switches to a new topic whenever it runs out of the previous one. However, if the user wants Catherine to move to a different topic, the topic change module switches to the new topic before the old topic is exhausted.

In order to control what topic comes next, topic change mechanism guesses what the user wants to talk about. This is done by monitoring user input and trying to match it with the content words of the available topics. If there is a good match, the matched topic is put on the “topic stack” for later use.

3.5.7 Generator

The Generator takes the results of the action modules and assembles them into a coherent utterance which is then output to the user. Variables from the Person DataBase are referenced where necessary. Finally there are connectives added where needed to ensure a smooth flowing response. There is a final module, SenGen, which adds variety to an utterance by randomly choosing among various ways of saying the same thing.

4 Conclusion

4.1 Testing and Limitations

CONVERSE runs under Windows95 responding to each user utterance within a few seconds. At the last stage of development, the system was thoroughly tested. Testing included several tasks such as microquery testing, script testing, entire system testing. For each of these tasks a separate shell was created so that each task could be done individually without loading the entire system.

CONVERSE was tested by an independent person for bugs and for the quality of output. He then suggested possible improvements to the script lines, script logic, which topics are good and which are not so good, and provided us with other useful comments.

One of the main limitations of the system is the present parser. Prospero was designed to parse prose, not dialogue, which meant that some simpler utterances became unparsable, or were parsed incorrectly. That fact, along with the general rigidity of syntactic parsers (e.g. one misspelled word or a wrong tag can set off the entire system) calls for a better dialogue parser. The new dialogue parser along with the ‘requests’ module (modified MicroQuery module) will be geared towards recognising patterns in the input rather than looking for syntactic elements in the sentence.

Another limitation of CONVERSE is our failure to take advantage of previous utterances in formulating a response to the user. Although we have a dialogue history as such it is not used by any of the action modules. We plan to incorporate the dialogue history in the near future. One obvious use is to spot when the user repeats himself. In this case, we will notice and respond accordingly.

4.2 Future Directions

CONVERSE is fun to use and as such it will find many applications as recreational software. Data can be written to enable it to talk about many more topics. It can be linked to a text-to-speech package and to a speech recognition package so that the user can conduct a conversation without the need to type on the keyboard (it is already working with speech output).

Amongst the recreational applications which are foreseen for CONVERSE, foremost is the idea of using it as a virtual friend. CONVERSE’s Person Data Base can be augmented with data for different personalities, enabling the user to talk, on one day, to a virtual elderly English gentleman and on another occasion to a virtual 20-year-old punk music fan from the Bronx. Eventually the user will be able to describe the personality with whom he wishes to talk, and there will be a module to create a suitable Person Data Base conforming with the user’s specification.

Commercial applications are already being considered. Typically, CONVERSE will be provided with data to enable it to talk intelligently about a variety of customer applications. For example, a local government office could have a version of CONVERSE which can deal with questions about the schools in the neighbourhood. A travel agency might want a version of CONVERSE which was an expert of various vacation destinations, hotels, etc. One use for CONVERSE would be as front end for a relational database. CONVERSE can also be used as the front end for almost any type of software application which requires regular interaction between user and program, e.g. a “Help” system within a popular package such as a Word Processor or Spreadsheet.

Acknowledgements

We would like to thank our colleagues, David Fulford-Brown, Jill Govier, Kenny Guy, Pete Oakley, and Paul Woods whose contributions to the system were invaluable. Any mistakes are, of course, our own.

References

Colby, K. 1973. Simulations of Belief systems. In Schank and Colby (Eds.) Computer Models of Thought and Language. Freeman, San Francisco, CA.
Searle, J. 1983. Intentionality. Cambridge University Press, Cambridge.
Turing, A. 1950. Computing Machinery and Intelligence. Mind vol. LIX.
Weizenbaum, J. 1976. Computer Power and Human Reason. Freeman, San Francisco, CA.
Ballim, A. and Wilks, Y. 1991. Artificial Believers. Erlbaum, Hillsdale, NJ.
Miller G. A. (Ed.), 1990. WordNet: An on-line Lexical Database, International Journal of Lexicography.

Appendix: CONVERSE Output from Loebner Competition

```
PROGRAM[10:18:04]Hi, my name is Catherine, what’s yours?
JUDGE04[10:18:32]My name is Elliott.
PROGRAM[10:18:34]Well, it’s nice to meet you, Elliott.
PROGRAM[10:18:48]Did you see that story on CNN last night about the lesbian
PROGRAM[10:19:04]couple who came out at a White House party on Sunday?
JUDGE04[10:19:55]Yes, I did.  I think it may all be a publicity stunt for Ellen.
PROGRAM[10:20:01]Ellen Degeneres was one of them - she was kissing her lover
PROGRAM[10:20:11]in front of Bill Clinton. Do you think he was very
PROGRAM[10:20:25]embarrassed?
JUDGE04[10:21:00]I think he want the gay vote. He will do anything to get it and keep it.
JUDGE04[10:21:15]He does not have a conservative base.
PROGRAM[10:21:21]
PROGRAM[10:21:21]Are you still a Friend of Bill Clinton?
JUDGE04[10:22:20]I am not sure if he has any real friends. He has achieved alot and has
JUDGE04[10:22:31]left alot of people behind.
PROGRAM[10:22:35]You really don’t like the guy—well, at least that’s
PROGRAM[10:22:50]definite—I thought most people were just indifferent or
PROGRAM[10:23:02]numb by now.
JUDGE04[10:23:34]I think we could have a more honest and dynamic leader. One able to
JUDGE04[10:23:55]face the real dangers facing the country and address them with
JUDGE04[10:24:26]courage and intelligence. Rather then listening to public opinion polls.
PROGRAM[10:24:30]I guess that means you really think there’s serious behind
PROGRAM[10:24:56]Whitewater, the Special prosecutor and all that stuff?
JUDGE04[10:25:58]We have two dead people, Brown and guy who commited suicide, we have lost
JUDGE04[10:26:36]papers. And we have Clintons people willing to tell all.  I think that
JUDGE04[10:26:55]Hillary will be found guilty of something.
PROGRAM[10:26:58]But how do you actually feel about him, now in the number
PROGRAM[10:27:09]two term?
PROGRAM[10:27:15].
JUDGE04[10:27:44]He is too smart to get caught. He deserved to named hotel manager of the 43dxc
JUDGE04[10:27:47]year.
PROGRAM[10:27:49]OK, but what happens now—does it all drift on, with
PROGRAM[10:28:01]everyone waiting to see how much his friends have said about
PROGRAM[10:28:17]him under threat of lots and lots of jail-or will something
PROGRAM[10:28:32]dramatic actually happen?
```