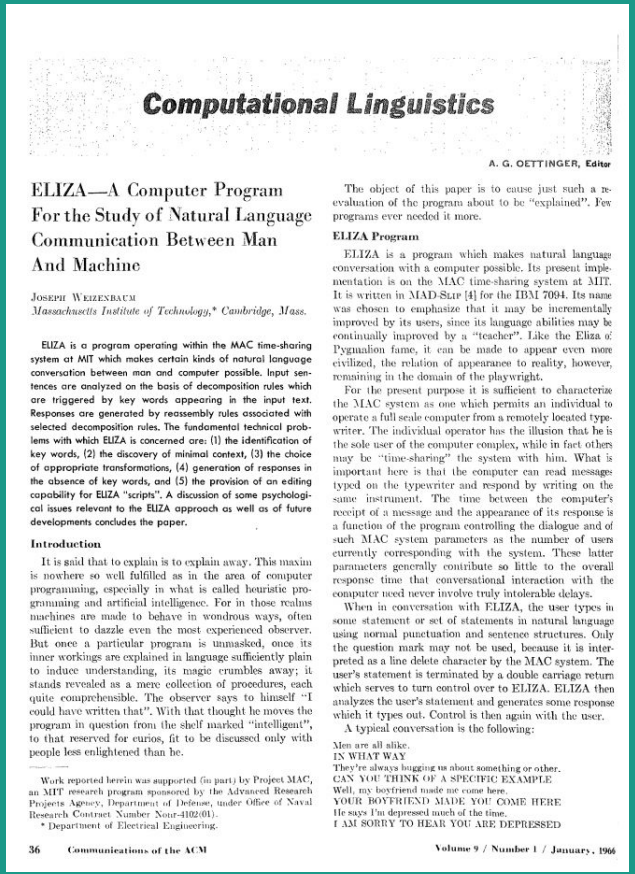
Rule Based Chatbot

Chaitanya Agarwal, Akshat Chhajer

Research



Computational Linguistics

A. G. OETTINGER, Editor

ELIZA—A Computer Program For the Study of Natural Language Communication Between Man And Machine

JOSEPH WEIZENBAUM
Massachusetts Institute of Technology, Cambridge, Mass.*

ELIZA is a program operating within the MAC time-sharing system at MIT which makes certain kinds of natural language conversation between man and computer possible. Input sentences are analyzed on the basis of decomposition rules which are triggered by key words appearing in the input text. Responses are generated by reassembly rules associated with selected decomposition rules. The fundamental technical problems with which ELIZA is concerned are: (1) the identification of key words, (2) the discovery of minimal context, (3) the choice of appropriate transformations, (4) generation of responses in the absence of key words, and (5) the provision of an editing capability for ELIZA "scripts". A discussion of some psychological issues relevant to the ELIZA approach as well as of future developments concludes the paper.

Introduction

It is said that to explain is to explain away. This maxim is nowhere so well fulfilled as in the area of computer programming, especially in what is called heuristic programming and artificial intelligence. For in those realms machines are made to behave in wondrous ways, often sufficient to dazzle even the most experienced observer. But once a particular program is unmasked, once its inner workings are explained in language sufficiently plain to induce understanding, its magic crumbles away; it stands revealed as a mere collection of procedures, each quite comprehensible. The observer says to himself "I could have written that". With that thought he moves the program in question from the shelf marked "intelligent", to that reserved for curios, fit to be discussed only with people less enlightened than he.

Work reported herein was supported (in part) by Project MAC, an MIT research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number Nonr-4102(01).

* Department of Electrical Engineering.

The object of this paper is to cause just such a reevaluation of the program about to be "explained". Few programs ever needed it more.

ELIZA Program

ELIZA is a program which makes natural language conversation with a computer possible. Its present implementation is on the MAC time-sharing system at MIT. It is written in MAD-SLIP [4] for the IBM 7094. Its name was chosen to emphasize that it may be incrementally improved by its users, since its language abilities may be continually improved by a "teacher". Like the Eliza of Pygmalion fame, it can be made to appear even more civilized, the relation of appearance to reality, however, remaining in the domain of the playwright.

For the present purpose it is sufficient to characterize the MAC system as one which permits an individual to operate a full scale computer from a remotely located typewriter. The individual operator has the illusion that he is the sole user of the computer complex, while in fact others may be "time-sharing" the system with him. What is important here is that the computer can read messages typed on the typewriter and respond by writing on the same instrument. The time between the computer's receipt of a message and the appearance of its response is a function of the program controlling the dialogue and of such MAC system parameters as the number of users currently corresponding with the system. These latter parameters generally contribute so little to the overall response time that conversational interaction with the computer need never involve truly intolerable delays.

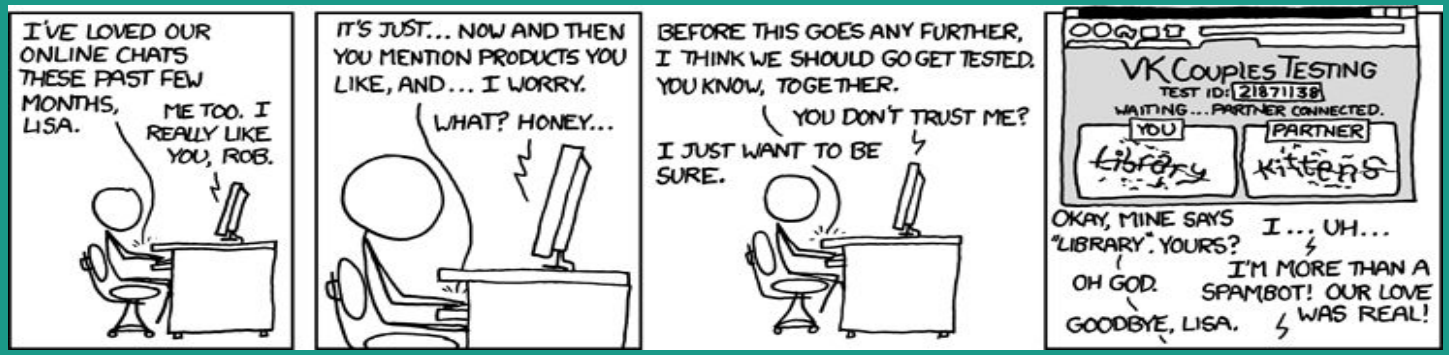
When in conversation with ELIZA, the user types in some statement or set of statements in natural language using normal punctuation and sentence structures. Only the question mark may not be used, because it is interpreted as a line delete character by the MAC system. The user's statement is terminated by a double carriage return which serves to turn control over to ELIZA. ELIZA then analyzes the user's statement and generates some response which it types out. Control is then again with the user.

A typical conversation is the following:

Men are all alike.
IN WHAT WAY
They're always bugging us about something or other.
CAN YOU THINK OF A SPECIFIC EXAMPLE
Well, my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE
He says I'm depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED

Preliminary Implementation

ELIZA IN ENGLISH



```
gReflections = {
  "am"   : "are",
  "was"  : "were",
  "i"    : "you",
  "i'd"  : "you would",
  "i've"  : "you have",
  "i'll"  : "you will",
  "my"  : "your",
  "are"  : "am",
  "you've": "I have",
  "you'll": "I will",
  "your"  : "my",
  "yours"  : "mine",
  "you"  : "me",
  "me"  : "you"
}
```

```
[r'Hello(.*)',
[  "Hello... I'm glad you could come today.",
  "Hi there... how are you today?",
  "Hello, how are you feeling today?"]],

[r'I think (.*)',
[  "Do you doubt %1?",
  "Do you really think so?",
  "But you're not sure %1?"]],
```

```
[r'(.*)',
[  "Please tell me more.",
  "Can you elaborate on that?",
  "Why do you say that %1?",
  "I see.",
  "%1.",
  "I didnt understand that."
]
```

Hindi ELIZA

Design Paradigm

```
class eliza:
  def __init__(self):
    self.keys = list(map(lambda x:re.compile(x[0], re.IGNORECASE),gPats))
    self.values = list(map(lambda x:x[1],gPats))

  def translate(self,str,dict):
    words = str.lower().split()
    keys = dict.keys();
    for i in range(0,len(words)):
      if words[i] in keys:
        words[i] = dict[words[i]]
    return ' '.join(words)

  def respond(self,str):
    # find a match among keys
    for i in range(0, len(self.keys)):
      match = self.keys[i].match(str)
      if match:
        # found a match ... stuff with corresponding value
        # chosen randomly from among the available options
        resp = random.choice(self.values[i])
        # we've got a response... stuff in reflected text where indicated
        pos = resp.find('%')
        while pos > -1:
          num = int(resp[pos+1:pos+2])
          resp = resp[:pos] + \
            self.translate(match.group(num),gReflections) + \
            resp[pos+2:]
          pos = resp.find('%')
        return resp
```

```
[r'मेरा|मेरी (.*)',
[  "अच्छा, आपका %1.",
  "आप ऐसा क्यों कह रहे की आपका %1?"]],
```

```
[r'अलविदा',
[  "मुझ से बात करने के लिए शुक्रिया।",
  "अलविदा",
  "धन्यवाद। आपका १५००० रूपए का बिल है।"]],
```

```
[r'मुझे (.*) चाहिए (.*)',
[  "आपको %1 क्यों चाहिए ?",
  "क्या आपको %1 से मदद मिलेगी??",
  "क्या आपको यकीन है आपको %1 चाहिए??"]],
```

```
[r'(.*)\?',
[  "आप ऐसा क्यों पूछ रहे है?",
  "इस सवाल का जवाब आप ही क्यों नहीं देते?"]],
```

Results

```
Therapist
---------
Talk to the program by typing in plain Hindi.
Enter "अलविदा" when done.
==========================================================================
नमस्ते! आप आज कैसा महसूस कर रहे हैं?
> नमस्ते
नमस्ते, आज आप कैसे हो?
> मैं बीमार हूँ
मुझ इस बारे में और बताइये।
> मुझे आम चाहिए
क्या आपको यकीन है आपको आम चाहिए?
> हाँ
ठीक है। पर इस बारे में क्या आप और बता सकते है?
> क्या मेरा उपचार हो सकता है
बहुत ही रोचक।
> क्यूंकि यह गलत है
और क्या कारन आते है आपके दिमाग में?
> अलविदा
धन्यवाद। आपका १५००० रूपए का बिल है।
```

Thank You