ELIZA

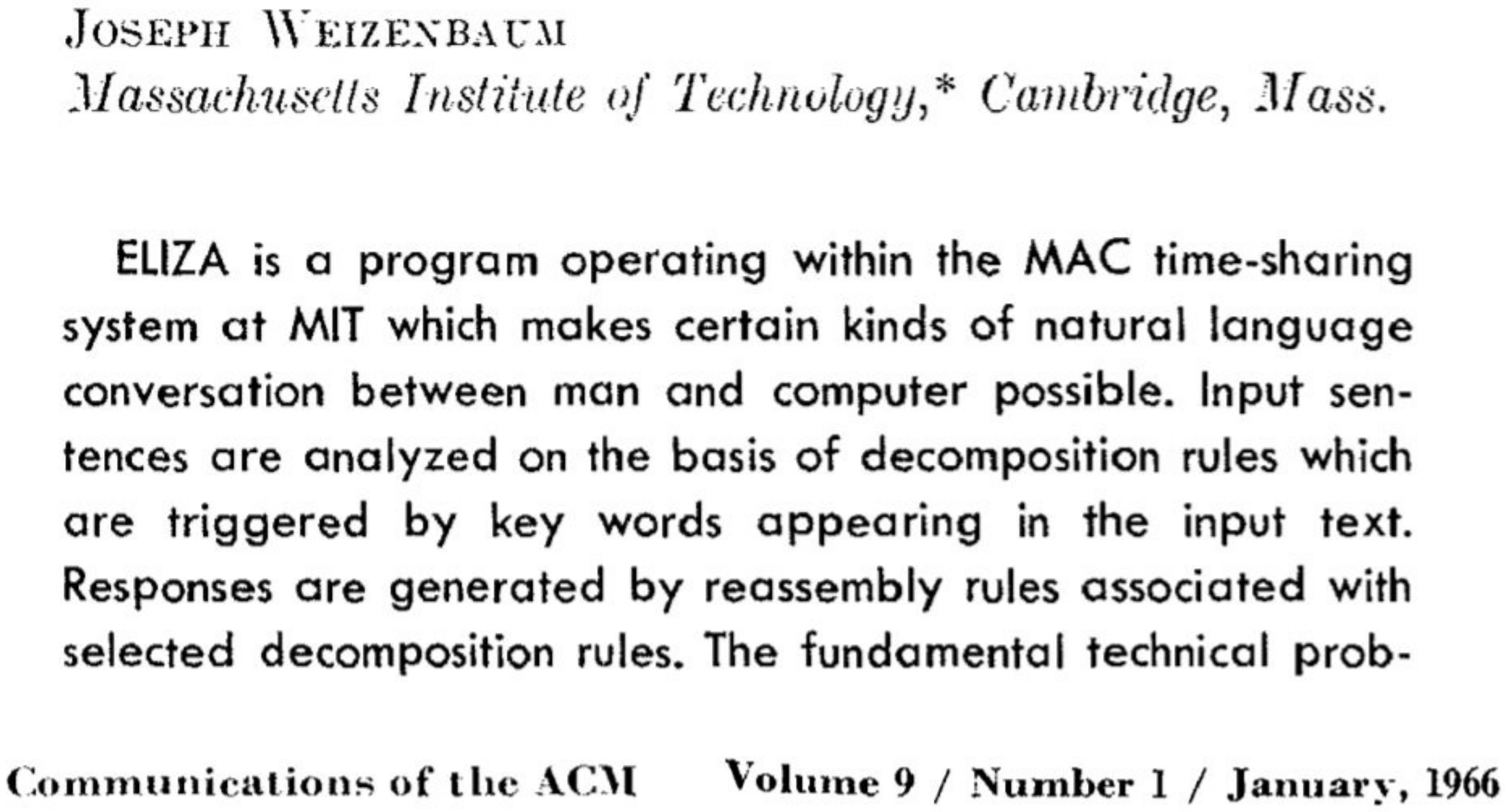
2019-04-11

ELIZA

*"A Computer Program For the **Study** of Natural Language Communication Between Man And Machine"*

OUR PLAN FOR TODAY

1) **SAY "*HI*" TO ELIZA**
2) **DOWNLOAD THE STARTER CODE**
3) **5 TASKS**
   Task specification by example
   Live demo
   Ideas for improvements
4) **CHALLENGES!**
   Breaking the code and debugging



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 prob-

Communications of the ACM   Volume 9 / Number 1 / January, 1966

Source: http://web.stanford.edu/class/cs124/p36-weizenabaum.pdf

TASK 0: Run chat

RUN ALL CELLS

In the end run chat function, type: “no” and press enter.

If Eliza answers: “Are you saying 'no' just to be negative?”

It means everything is OK.

TASK 1: Add responses

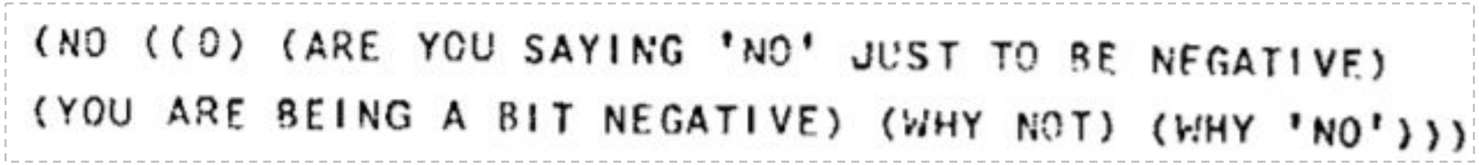
UNCOMMENT DOCTEST LINE FOR TASK 1:

doctest....task1

PASS THE TEST!

TASK 1: Add responses

- Inspect user input for the presence of the "**no**" keyword and respond if the keyword was found:
  *(**A**) "Are you saying 'no' just to be negative?", (**B**) "You are being a bit negative", (**C**) "Why not"*
- Cycle through responses: A, B, C, A, B...
- How Eliza does it?
    - process input word-by-word
    - keystack with list of found keywords



```
(NO ((0) (ARE YOU SAYING 'NO' JUST TO BE NEGATIVE)
(YOU ARE BEING A BIT NEGATIVE) (WHY NOT) (WHY 'NO')))
```

- How to split a sentence? "some utterance".split() <- splits by words
- Risk: random random
- Follow-up: preprocessing, text segmentation, turn-taking

TASK 1: Add responses

IMPROVEMENTS?

TASK 1: Add responses

IMPROVEMENTS?

- exception handling (**KeyboardInterrupt**)
- more keywords ("yes") (how much?)
- case sensitivity ("no" vs "NO") => normalization

TASK 2: Handle lack of keywords

UNCOMMENT DOCTEST LINE FOR TASK 2:

doctest….task2

PASS THE TEST!

TASK 2: Handle lack of keywords

IMPROVEMENTS?

TASK 2: Handle lack of keywords

IMPROVEMENTS?

- add responses for "none" + update code to rotate
- handle input without words ("?")

TASK 3: Add keywords and **substitutions**

UNCOMMENT DOCTEST LINE FOR TASK 3:

doctest….task3

PASS THE TEST!

TASK 3: Add keywords and **substitutions**

- **Example decomposition and reassembly:**

```
>>> utt = "I am learning about Eliza"
>>> decomposition = r".*\bI am (.*)"
>>> reassembly = r"Is it because you are \1 that you.."
# sub method
>>> regex.sub(decomposition, reassembly, utt)
Is it because you are learning about Eliza that you...
```

- **Note on regular expressions:**
  .* - matches *anything (.) of any length (*)*
  \b - matches word **b**oundary

Substitutions:

"yourself" => "myself",
"me" => "you",
"I" => "you",
"am" => "are",
"you" => "I",
"are" => "am",
"my" => "your"

TASK 3: Add keywords and **substitutions**

IMPROVEMENTS?

TASK 3: Add keywords and **substitutions**

IMPROVEMENTS?

- "I am" vs "I'm", "don't" vs "do not" => normalization
- what about Polish morphology?
- multiple decomposition/reassembly rules per keyword

TASK 4: Add **MORE** keywords and substitutions

UNCOMMENT DOCTEST LINE FOR TASK 4:

doctest….task4

TASK 4: Add **MORE** keywords and substitutions

IMPROVEMENTS?

TASK 4: Add **MORE** keywords and substitutions

IMPROVEMENTS?

- punctuation ("NO!") => tokenization and segmentation
- segmentation ("I said no. You said yes.")
- lemmatization/stemming ("nooo", "students" vs "student")
- session object vs in-memory script modification

TASK 5: Add memory

UNCOMMENT DOCTEST LINE FOR TASK 5:

doctest….task5

PASS THE TEST!

TASK 5: Add memory

IMPROVEMENTS?

TASK 5: Add memory

IMPROVEMENTS?

- number of words as part of the transformation rules?
- refactor code (update design based on current flow/requirements)
- separate script ("knowledge base") and code - simplify script format
- replace complex doctest with dedicated dialogue scenario tests ("unit tests")
- create more patterns for "memory"
- multiple decomposition/reassembly rules per keyword with different ranks
- handle more than 1 transformation for memory based responses

TASK 6: EXTRA

Demonstrate the effect of "**rank**" in script.

YOUR TURN!

CHALLENGES!

Breaking the code and debugging

FOOLY CONNECTED

THE END

"I am not sure if I understand you fully"