SHRDLU

SHRDLU was an **early natural language understanding program**. It was developed by Terry Winograd at MIT in 1968-1979. In it, the user carries on a conversation with the computer, moving objects, naming collections and querying the state of a simplified '**blocks world**', essentially a virtual box filled with different blocks. For example:

> Person: *'Pick up a big red block'*
> SHRDLU: '*OK.' (Picks up the block in the display using its animated hand.)*
> Person: *'Grasp the pyramid'*
> SHRDLU: *'I DON'T UNDERSTAND WHICH PYRAMID YOU MEAN'*
> Person: *'Find a block which is taller than the one you are holding and put it into the box.'*
> SHRDLU: *'By "IT", I ASSUME YOU MEAN THE BLOCK WHICH IS TALLER THAN THE ONE I AM HOLDING.'*
> SHRDLU: *'OK.' (Does it.)*

SHRDLU had to **discover sequences of basic actions** that would achieve its goals. It did this with a **'Planner'** program. It used a technique now called **backward chaining** or **means-ends analysis** which checked for every presupposition of the given task to be true. For example, if the task was to have *'a red block on the green block',* the Planner would first check if it were already true, and if not, it would try to find a way to achieve it.
To find a way, it would search through the various operations in its **knowledge base**. In the previous example (*'a red block on the green block'*), this would yield *UNGRASP* as a possible way to satisfy the goal, provided that three preconditions held: (1) the green block had nothing on top, (2) the hand was holding a red block, and (3) the hand was over the green block. The Planner would then check to see if these preconditions were true, and if not, develop **sub-plans** to achieve each precondition.

The Planner used a **special internal representation**. The proposition, *'a red block in the box'* would be represented as something like:

> (**THGOAL** (**#IS** ?X **#BLOCK**))
> (**THGOAL** (**#COLOR** $?X **#RED**))
> (**THGOAL** (**#IN** $?X :**BOX**))

Since SHRDLU's inputs were English sentences, it first had to **parse** them - to work out where the noun phrases began and ended, which words went with which, and so on. SHRDLU embodies a highly ***'interactionist'*** view of the relation between syntactic and semantic processing. For example, consider the command '*Put the blue pyramid on the block in the box*' which is ambiguous (as to whether '*on the block*' describes the current location of the pyramid or its desired location). In this case, SHRDLU would first recognize that '*the blue pyramid*' was a syntactically possible noun phrase, then immediately check to see if this interpretation made

sense semantically. If so, the parser would know that the rest of the sentence, '*on the block in the box'*, had to be a single prepositional phrase, specifying where to put the pyramid. If, one the other hand, there was more than one blue pyramid, the parser would deduce that '*on the block*' was part of the noun phrase; then SHRDLU would perform another semantic check, to make sure that '*the blue pyramid on the block*' was a valid description of a unique object.

SHRDLU could hold a reasonably involved conversation and not lose the **thread** - something that is not trivial even for people. For example, in the dialogue above SHRDLU deals with words like '*it'* and '*that'* which refer back to things it heard or did or said earlier. It used a **back-tracking** method to **map ellided information to symbols**, where it considered the **head** of the noun phrase in the previous statement as the **resolution**.

ELIZA

ELIZA (Weizenbaum, 1966) is the most important chatbot dialogue system in the history of the field of **Converational AIs**. ELIZA was designed to simulate a **Rogerian psychologist**, based on a branch of clinical psychology whose methods involve drawing the patient out by reflecting patient's statements back at them. Rogerian psychology is the rare type of conversation in which, as Weizenbaum points out, one can "assume the pose of knowing almost nothing of the real world". If a patient says "I went for a long boat ride" and the psychiatrist says "Tell me about boats", you don't assume she didn't know what a boat is, but rather assume she had some conversational goal. Most chatbots trying to pass the Turing test choose a domain with similar properties. Consider the following conversation where ELIZA tries to give very therapeutic advice, exercepted from a full interaction given in the original paper.

Person: Men are all alike.
ELIZA: IN WHAT WAY
Person: They're always bugging us about something or other.
ELIZA: CAN YOU THINK OF A SPECIFIC EXAMPLE
Person: Well my boyfriend made me come here.
ELIZA: YOUR BOYFRIEND MADE YOU COME HERE
Person: He says I'm depressed much of the time.
ELIZA: I AM SORRY TO HEAR THAT YOU ARE DEPRESSED
..
..
ELIZA: WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU
Person: My father
ELIZA: YOUR FATHER
Person: You are like my father in some ways.
ELIZA: WHAT RESEMBLANCE DO YOU SEE
Person: You are not very aggressive but I think you don't want me to notice that.
ELIZA: WHAT MAKES YOU THINK I AM NOT AGGRESSIVE
Person: You are afraid of me
ELIZA: DOES IT PLEASE YOU TO BELIEVE I'M AFRAID OF YOU
Person: My father is afraid of everybody
ELIZA: WHAT ELSE COMES TO YOUR MIND WHEN YOU THINK OF YOUR FATHER
Person: Bullies.
ELIZA: DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR BOYFRIEND MADE YOU COME HERE

ELIZA works by **pattern/transform rule matching**. Consider the following:

($ YOU $ ME) [pattern] -> (WHAT MAKES YOU THINK I $ YOU) [transform]

This rule would transfer *You hate me* into *WHAT MAKES YOU THINK I HATE YOU*.

Each ELIZA pattern/rule is linked to a keyword that might occur in a user input. Keywords are associated with a rank, with specific words being more highly ranked, and more general words ranking lower. Consider the following user sentence:

I know everybody laughed at me

Because it has the word "I", this sentence could match the following rule whose keyword is *I:*

(I *) -> (YOU SAY YOU 2)

Producing:

YOU SAY YOU KNOW EVERYBODY LAUGHED AT YOU

If no keyword matches, ELIZA chooses a non-committal response like *PLEASE GO ON, THAT'S VERY INTERESTING* or *I SEE.*

The **algorithm** that drives it:

function ELIZA GENERATOR(user *sentence*) **returns** *response*
Find the word w in *sentence* that has the highest keyword rank
if w exists
Choose the highest ranked rule r for w that matches *sentence*
response←Apply the transform in r to *sentence*
if w = 'my'
future←Apply a transformation from the 'memory' rule list to *sentence*
Push future onto memory stack
else (no keyword applies)
either response←Apply the transform for the NONE keyword to sentence
or
response←Pop the top response from the memory stack
return(*response*)