Regular Expressions

Rule-based vs Statistical Approaches

Rule-based = linguistic

Statistical = “learn” from a large corpus that has been marked up with the phenomena you are studying (Machine Learning)

For what problems is rule-based better suited and when is statistical better?

Depends on the problem: classification/categorization problem?

Depends on your resources: how much good training data is available?

Even when using statistical – many tasks easily done with rules: tokenization, sentence breaking, morphology

Some PARTS of a task may be done with rules: e.g., rules may be used to extract features, and then statistical learning methods might combine those features to perform a task.

Today we are going to …

- Review Regular Expressions
  - Allows one to capture patterns in text to develop rule-based methods or use to collect features for stochastic methods (machine learning; statistical methods)
  - There is an old joke:

    Some people, when confronted with a problem, think

    "I know, I'll use regular expressions."

    Now they have two problems. – Jamie Zawinski
  - Regular Expressions are handy and allow us to create complex rules to capture patterns in text

Regular Expressions

- Can be viewed as a way to specify:
  - Search patterns over text string
  - Design of a particular kind of machine, called a Finite State Automaton (FSA)
- These are really equivalent

Uses of Regular Expressions in NLP

- As grep, perl: Simple but powerful tools for large corpus analysis and 'shallow' processing
  - What word is most likely to begin a sentence?
  - What word is most likely to begin a question?
  - In your own email, are you more or less polite than the people you correspond with?

- With other unix tools, allow us to
  - Obtain word frequency and co-occurrence statistics
  - Build simple interactive applications

- Regular expressions define regular languages or sets

Some definitions

- **Regular Expression**: Formula in algebraic notation for specifying a set of strings

- **String**: Any sequence of alphanumeric characters
  - Letters, numbers, spaces, tabs, punctuation marks

- **Regular Expression Search**
  - Pattern: specifying the set of strings we want to search for
  - Corpus: the texts we want to search through

Simple Example

RE	DESCRIPTION	USES
/This/	Matches the string “This”	Finding the word “this” starting a sentence
/this/	Matches the string “this”	Finding the word “this” internal to a sentence.
/[Tt]his/	Matches either “This” or “this” -- disjunction	Finding the word “this” anywhere in a sentence.

Some regexes

Regex	Description
/./	Wild card; any character
/\./	Period
/a/	Any ‘a’
/[ab]/	Any a or b (choice)
/(ab)/	The string ab
/[a-z]/	Any lowercase character (range)
/[A-Z]/	Any upper case character (range)
/[^?.!]/	Any non ?, . or ! (negation of set)
/\s/	White space

More regexes

Regex	Example	Description
*	/a*/	Zero or more a’s
+	/a+/	One or more a’s
?	/a?/	Zero or one a
|	/cat|dog/	The strings cat or dog
\b	/\bthe\b/	The word ‘the’
\B	\bun\B	Words prefixed by ‘un’; Beginning of a longer string
\1	\(again) and \1/	Using string captured by () in regex
$	/end of the line.$/	Denotes end of a string
^	/^First word/	Denotes beginning of a string

Regex	Description
/./	Wild card; any character
/\./	Period
/a/	Any 'a'
/[ab]/	Any a or b (choice)
/(ab)/	The string ab
/[a-z]/	Any lowercase character (range)
/[A-Z]/	Any upper case character (range)
/[^?.!]/	Any non ?, . or ! (negation of set)
/\s/	White space

Question

What is the regex to identify the word dog and its plural form?

Regex	Example	Description
*	/a*/	Zero or more a's
+	/a+/	One or more a's
?	/a?/	Zero or one a
|	/cat|dog/	The strings cat or dog
\b	/\bthe\b/	The word 'the'
\B	\bun\B	Words prefixed by 'un'; Beginning of a longer string
\1	\(again) and \1/	Using string captured by () in regex
$	/end of the line.$/	Denotes end of a string
^	/^First word/	Denotes beginning of a string

Regex	Description
/./	Wild card; any character
/\./	Period
/a/	Any 'a'
/[ab]/	Any a or b (choice)
/(ab)/	The string ab
/[a-z]/	Any lowercase character (range)
/[A-Z]/	Any upper case character (range)
/[^?.!]/	Any non ?, . or ! (negation of set)
/\s/	White space

Answer

/\bdogs?\b/

Regex	Example	Description
*	/a*/	Zero or more a's
+	/a+/	One or more a's
?	/a?/	Zero or one a
|	/cat|dog/	The strings cat or dog
\b	/\bthe\b/	The word 'the'
\B	\bun\B	Words prefixed by 'un'; Beginning of a longer string
\1	\(again) and \1/	Using string captured by () in regex
$	/end of the line.$/	Denotes end of a string
^	/^First word/	Denotes beginning of a string

Regex	Description
/./	Wild card; any character
/\./	Period
/a/	Any ‘a’
/[ab]/	Any a or b (choice)
/(ab)/	The string ab
/[a-z]/	Any lowercase character (range)
/[A-Z]/	Any upper case character (range)
/[^?.!]/	Any non ?, . or ! (negation of set)
/\s/	White space

Question

What is the regex to identify the word puppy and its plural form?

Regex	Example	Description
*	/a*/	Zero or more a’s
+	/a+/	One or more a’s
?	/a?/	Zero or one a
|	/cat|dog/	The strings cat or dog
\b	/\bthe\b/	The word ‘the’
\B	\bun\B	Words prefixed by ‘un’; Beginning of a longer string
\1	\(again) and \1/	Using string captured by () in regex
$	/end of the line.$/	Denotes end of a string
^	/^First word/	Denotes beginning of a string

Regex	Description
/./	Wild card; any character
/\./	Period
/a/	Any 'a'
/[ab]/	Any a or b (choice)
/(ab)/	The string ab
/[a-z]/	Any lowercase character (range)
/[A-Z]/	Any upper case character (range)
/[^?.!]/	Any non ?, . or ! (negation of set)
/\s/	White space

Answer

/\bpupp(y|ies)\b/

Regex	Example	Description
*	/a*/	Zero or more a's
+	/a+/	One or more a's
?	/a?/	Zero or one a
|	/cat|dog/	The strings cat or dog
\b	/\bthe\b/	The word 'the'
\B	\bun\B	Words prefixed by 'un'; Beginning of a longer string
\1	\(again) and \1/	Using string captured by () in regex
$	/end of the line.$/	Denotes end of a string
^	/^First word/	Denotes beginning of a string

Regex	Description
/./	Wild card; any character
/\./	Period
/a/	Any 'a'
/[ab]/	Any a or b (choice)
/(ab)/	The string ab
/[a-z]/	Any lowercase character (range)
/[A-Z]/	Any upper case character (range)
/[^?.!]/	Any non ?, . or ! (negation of set)
/\s/	White space

Question?

- What would my regular expression look like if I wanted to only match:
  - Temperature and TeMPerature

Regex	Example	Description
*	/a*/	Zero or more a's
+	/a+/	One or more a's
?	/a?/	Zero or one a
|	/cat|dog/	The strings cat or dog
\b	/\bthe\b/	The word 'the'
\B	\bun\B	Words prefixed by 'un'; Beginning of a longer string
\1	\(again) and \1/	Using string captured by () in regex
$	/end of the line.$/	Denotes end of a string
^	/^First word/	Denotes beginning of a string

Regex	Description
/./	Wild card; any character
/\./	Period
/a/	Any 'a'
/[ab]/	Any a or b (choice)
/(ab)/	The string ab
/[a-z]/	Any lowercase character (range)
/[A-Z]/	Any upper case character (range)
/[^?.!]/	Any non ?, . or ! (negation of set)
/\s/	White space

Answer

Te(MP|mp)erature

Regex	Example	Description
*	/a*/	Zero or more a's
+	/a+/	One or more a's
?	/a?/	Zero or one a
|	/cat|dog/	The strings cat or dog
\b	/\bthe\b/	The word 'the'
\B	\bun\B	Words prefixed by 'un'; Beginning of a longer string
\1	\(again) and \1/	Using string captured by () in regex
$	/end of the line.$/	Denotes end of a string
^	/^First word/	Denotes beginning of a string

Regex	Description
/./	Wild card; any character
/\./	Period
/a/	Any 'a'
/[ab]/	Any a or b (choice)
/(ab)/	The string ab
/[a-z]/	Any lowercase character (range)
/[A-Z]/	Any upper case character (range)
/[^?.!]/	Any non ?, . or ! (negation of set)
/\s/	White space

Question

What is the regex to identify all words that begin with no, or non, nonn, nonnn, etc. regardless of the n's?

Regex	Example	Description
*	/a*/	Zero or more a's
+	/a+/	One or more a's
?	/a?/	Zero or one a
|	/cat|dog/	The strings cat or dog
\b	/\bthe\b/	The word 'the'
\B	\bun\B	Words prefixed by 'un'; Beginning of a longer string
\1	\(again) and \1/	Using string captured by () in regex
$	/end of the line.$/	Denotes end of a string
^	/^First word/	Denotes beginning of a string

Answer

/\bnon*\B/

Regex	Description
/./	Wild card; any character
/\./	Period
/a/	Any 'a'
/[ab]/	Any a or b (choice)
/(ab)/	The string ab
/[a-z]/	Any lowercase character (range)
/[A-Z]/	Any upper case character (range)
/[^?.!]/	Any non ?, . or ! (negation of set)
/\s/	White space

Regex	Example	Description
*	/a*/	Zero or more a's
+	/a+/	One or more a's
?	/a?/	Zero or one a
|	/cat|dog/	The strings cat or dog
\b	/\bthe\b/	The word 'the'
\B	\bun\B	Words prefixed by 'un'; Beginning of a longer string
\1	\(again) and \1/	Using string captured by () in regex
$	/end of the line.$/	Denotes end of a string
^	/^First word/	Denotes beginning of a string

Regex	Description
/./	Wild card; any character
/\./	Period
/a/	Any 'a'
/[ab]/	Any a or b (choice)
/(ab)/	The string ab
/[a-z]/	Any lowercase character (range)
/[A-Z]/	Any upper case character (range)
/[^?.!]/	Any non ?, . or ! (negation of set)
/\s/	White space

Question

What is the regular expression to identify ier and ier phrases such as:

happier and happier

or

fuzzier and fuzzier

Regex	Example	Description
*	/a*/	Zero or more a's
+	/a+/	One or more a's
?	/a?/	Zero or one a
|	/cat|dog/	The strings cat or dog
\b	/\bthe\b/	The word 'the'
\B	\bun\B	Words prefixed by 'un'; Beginning of a longer string
\1	\(again) and \1/	Using string captured by () in regex
$	/end of the line.$/	Denotes end of a string
^	/^First word/	Denotes beginning of a string

Regex	Description
/./	Wild card; any character
/\./	Period
/a/	Any 'a'
/[ab]/	Any a or b (choice)
/(ab)/	The string ab
/[a-z]/	Any lowercase character (range)
/[A-Z]/	Any upper case character (range)
/[^?.!]/	Any non ?, . or ! (negation of set)
/\s/	White space

Answer

/(.+)ier and \1ier/

the \1 here is what ever was identified in the (.+)

Regex	Example	Description
*	/a*/	Zero or more a's
+	/a+/	One or more a's
?	/a?/	Zero or one a
|	/cat|dog/	The strings cat or dog
\b	/\bthe\b/	The word 'the'
\B	\bun\B	Words prefixed by 'un'; Beginning of a longer string
\1	\(again) and \1/	Using string captured by () in regex
$	/end of the line.$/	Denotes end of a string
^	/^First word/	Denotes beginning of a string

Optionality and Repetition

- /[Ww]oodchucks?/ matches woodchucks, Woodchucks, woodchuck, Woodchuck
- /colou?r/ matches color or colour
- /he{3}/ matches heee
- /(he){3}/ matches hehehe
- /(he){3,} matches a sequence of at least 3 he’s

Operator Precedence Hierarchy

1. Parentheses ()
2. Counters * + ? {}
3. Sequence of Anchors ^$
4. Disjunction |

/tr(y|ies)/

-> what is \1?

Operator Precedence Hierarchy

1. Parentheses ()
2. Counters * + ? {}
3. Sequence of Anchors ^$
4. Disjunction |

```
/tr(y|ies)/
          -> what is \1?

/^\s*$/
          -> what is this finding?
```

A Simple Exercise

- Write a regular expression to find all instances of the determiner "the":

<u>The</u> recent attempt by <u>the</u> police to retain their current rates of pay has not gathered much favor with <u>the</u> southern factions.

A Simple Exercise

- Write a regular expression to find all instances of the determiner “the”:

  /the/

The recent attempt by the police to retain their current rates of pay has not gathered much favor with the southern factions.

A Simple Exercise

- Write a regular expression to find all instances of the determiner "the":

  /the/

The recent attempt by the police to retain their current rates of pay has not gathered much favor with the southern factions.

A Simple Exercise

- Write a regular expression to find all instances of the determiner "the":

  ```
  /the/
  /\bthe\b/
  ```

The recent attempt by the police to retain their current rates of pay has not gathered much favor with the southern factions.

A Simple Exercise

- Write a regular expression to find all instances of the determiner "the":

  /the/

  /\bthe\b/

The recent attempt by <u>the</u> police to retain their current rates of pay has not gathered much favor with <u>the</u> southern factions.

A Simple Exercise

- Write a regular expression to find all instances of the determiner "the":

```
/the/
/[tT]he/
/\b[tT]he\b/
```

The recent attempt by the police to retain their current rates of pay has not gathered much favor with the southern factions.

A Simple Exercise

- Write a regular expression to find all instances of the determiner "the":

```
/the/
/[tT]he/
/\b[tT]he\b/
```

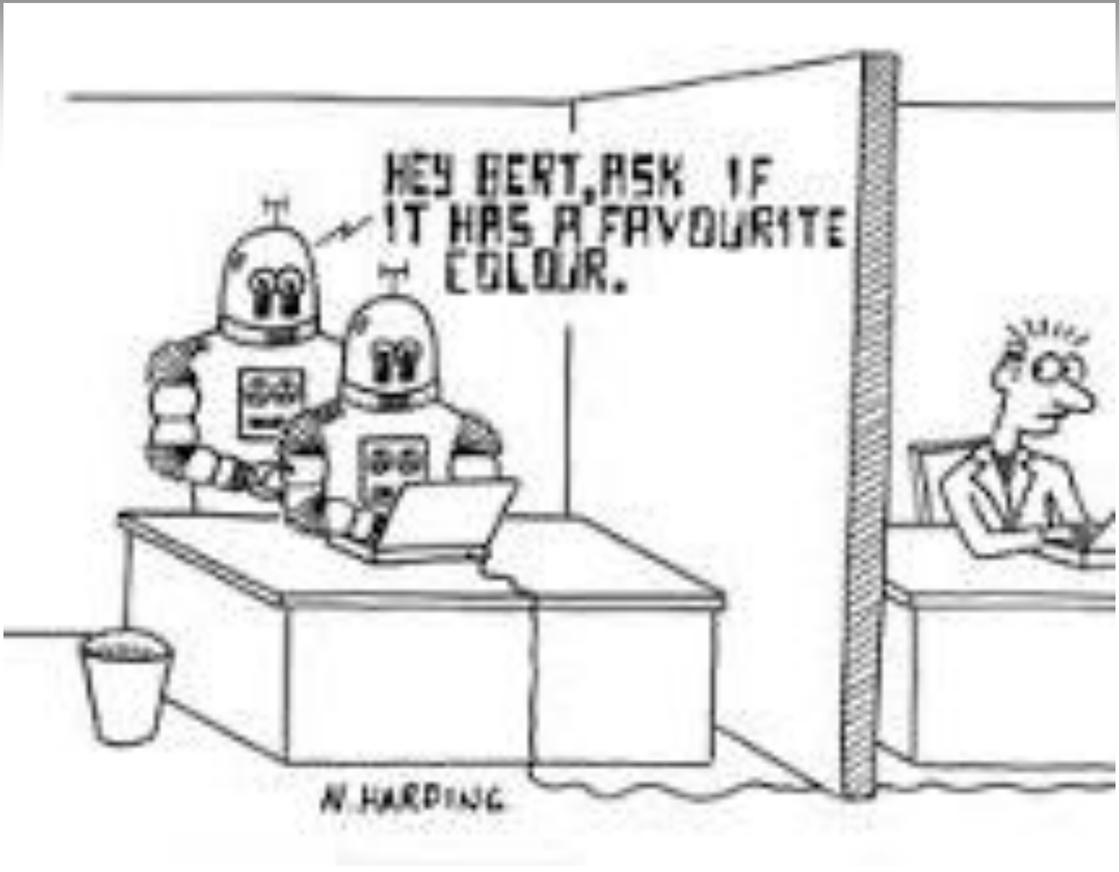
The recent attempt by the police to retain their current rates of pay has not gathered much favor with the southern factions.

Substitutions (Transductions)

- Sed or ‘s’ operator in Perl
  - s/regexp1/pattern/
  - s/I am feeling (.+)/You are feeling \1?/
    - I am feeling hungry -> You are feeling hungry?
  - s/I gave the (.+) to (.+)/Why would you give \2 \1?/
    - I gave the money to Brad -> Why would you give Brad money?

How does this all relate to language processing?

- aka why are regexes useful?
    - What do you do when you want to process each sentence in a paragraph individually and want it done quickly because you are receiving a new paragraph every second?
    - What happens if you want to obtain the number of times *immunosuppression* occurred in the text?

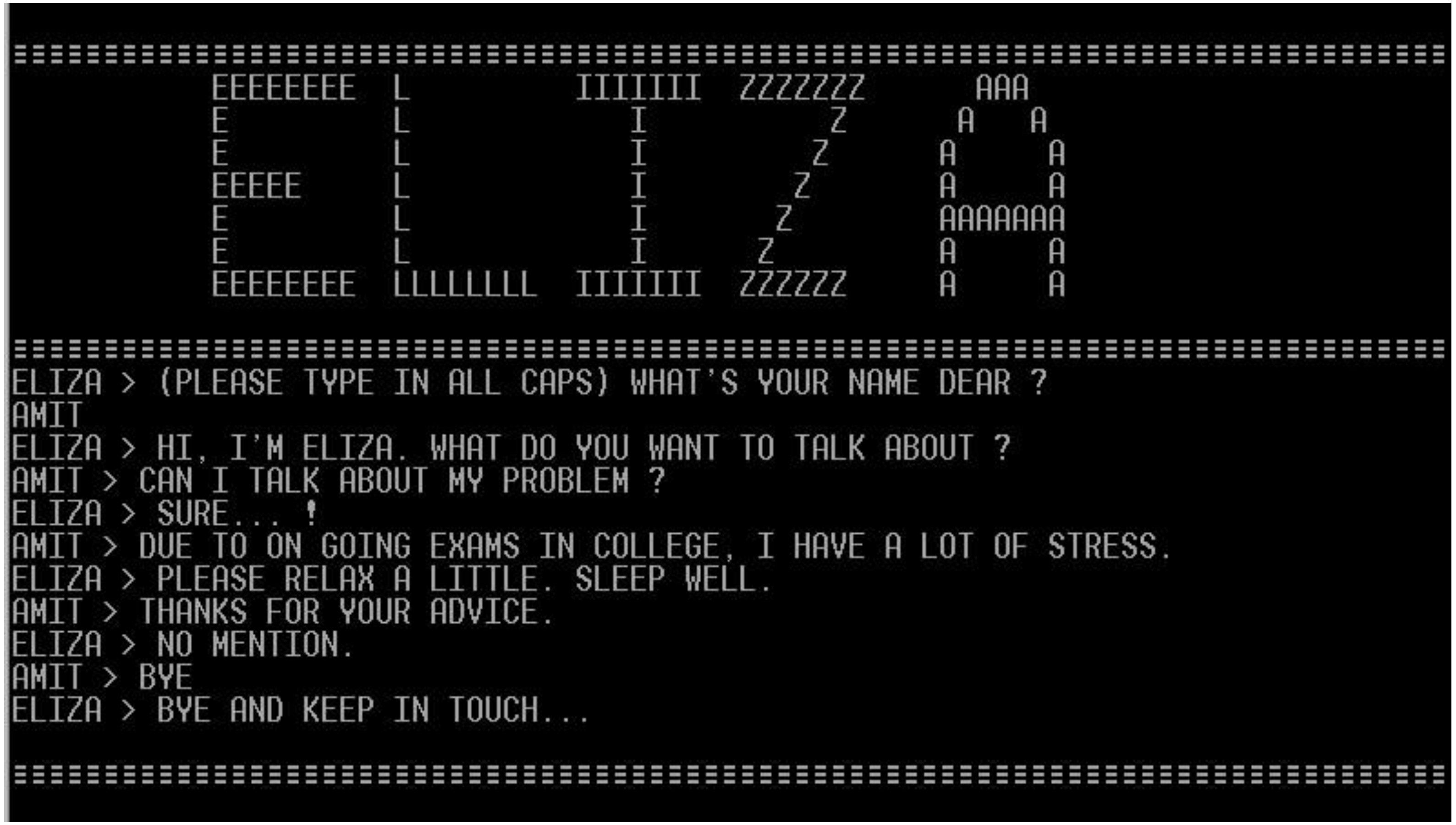


Computing Machinery and Intelligence

- Alan Turing
  - 1950:
    - published Computing Machinery and Intelligence
  - Asked the question:
    - how do we determine if a computer is intelligent
  - Proposed solution:
    - develop a computer program to impersonate a human in a real-time written conversation

Towards this end ....

1966: Joseph Weizenbaum developed Eliza
a Rogerian psychotherapist



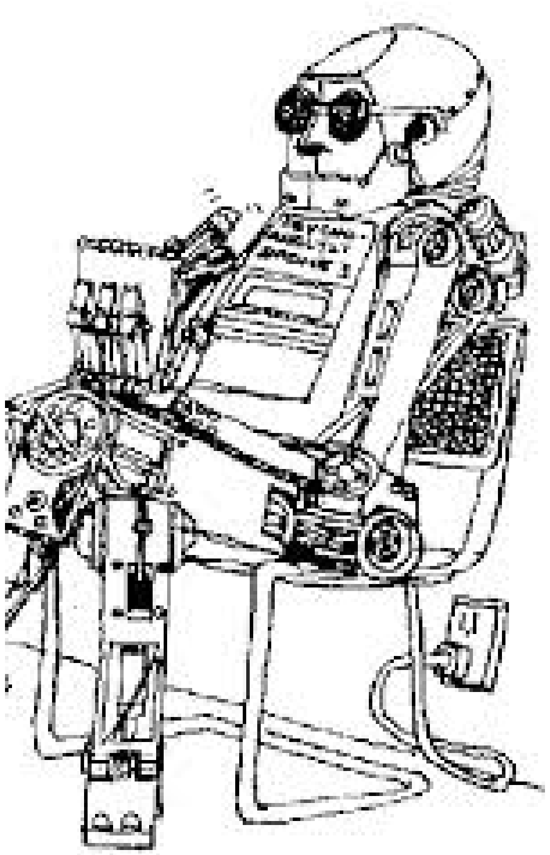
```
==========================================================================
          EEEEEEEE  L        IIIIIII  ZZZZZZZ      AAA
          E         L           I           Z     A   A
          E         L           I          Z     A     A
          EEEEE     L           I         Z      A     A
          E         L           I        Z       AAAAAAA
          E         L           I       Z        A     A
          EEEEEEEE  LLLLLLLL  IIIIIII  ZZZZZZ    A     A

==========================================================================
ELIZA > (PLEASE TYPE IN ALL CAPS) WHAT'S YOUR NAME DEAR ?
AMIT
ELIZA > HI, I'M ELIZA. WHAT DO YOU WANT TO TALK ABOUT ?
AMIT > CAN I TALK ABOUT MY PROBLEM ?
ELIZA > SURE... !
AMIT > DUE TO ON GOING EXAMS IN COLLEGE, I HAVE A LOT OF STRESS.
ELIZA > PLEASE RELAX A LITTLE. SLEEP WELL.
AMIT > THANKS FOR YOUR ADVICE.
ELIZA > NO MENTION.
AMIT > BYE
ELIZA > BYE AND KEEP IN TOUCH...

==========================================================================
```

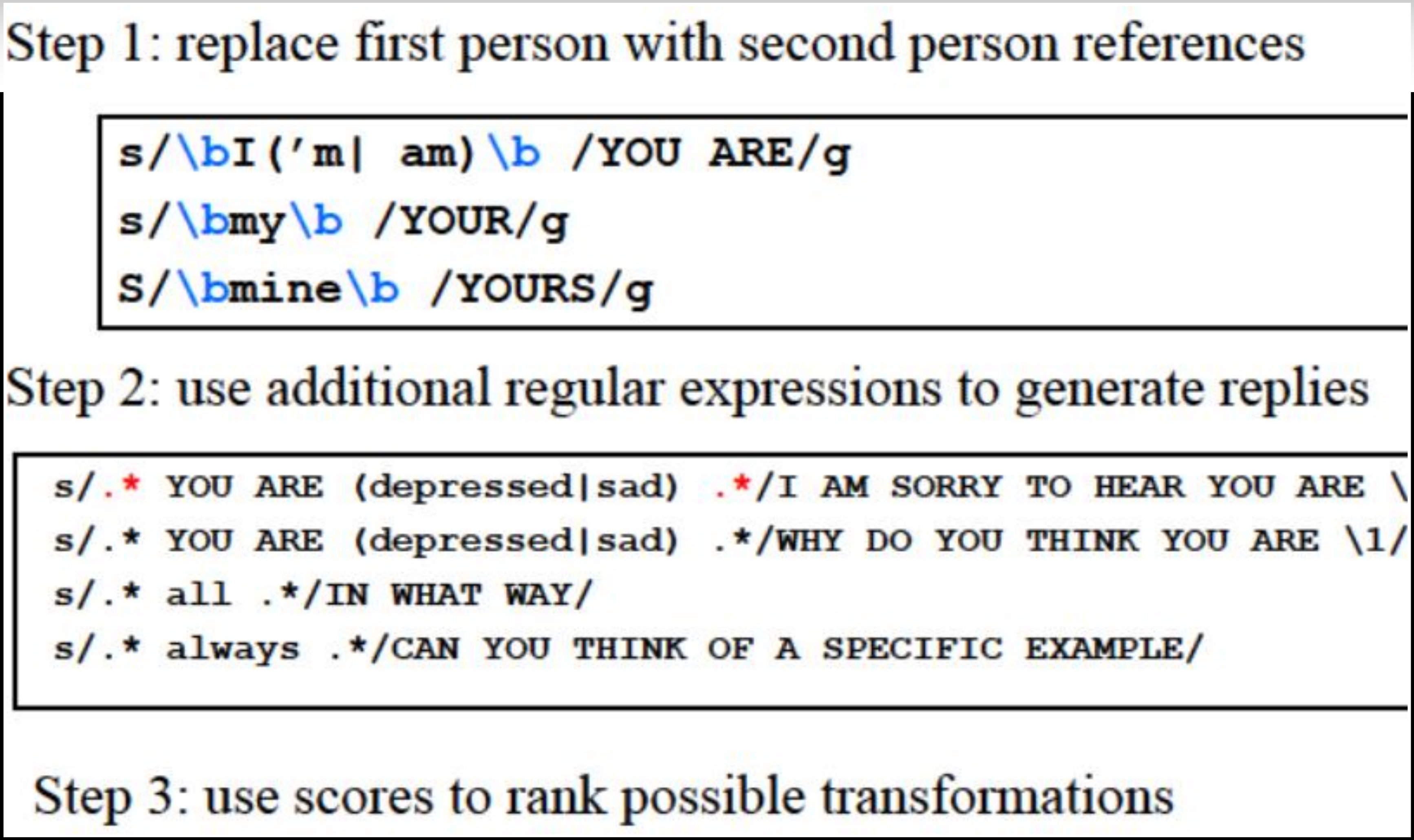
Eliza

- Simple program that uses pattern matching to process the input and translate it into suitable outputs.
- Succeeded because Eliza didn’t need to *know* anything in order to simulate a Rogerian psychotherapist.
  - Rogerian psychotherapy is a form of talk-psychotherapy that is person centered.
  - Weizenbaum notes: this is one of the few dialogue genres where listeners can act as if they know nothing of the world.
- Amazing thing: people believed that ELIZA understood them and their problems



Eliza (Weizenbaum)

- 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
- It's true. I am unhappy
- DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
- I need some help, that much seems certain.
- WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
- Perhaps I could learn to get along with my mother.
- TELL ME MORE ABOUT YOUR FAMILY
- My mother takes care of me.
- WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU
- My father.
- YOUR FATHER
- You are like my father in some ways.


Step 1: replace first person with second person references

```
s/\bI('m| am)\b /YOU ARE/g
s/\bmy\b /YOUR/g
S/\bmine\b /YOURS/g
```

Step 2: use additional regular expressions to generate replies

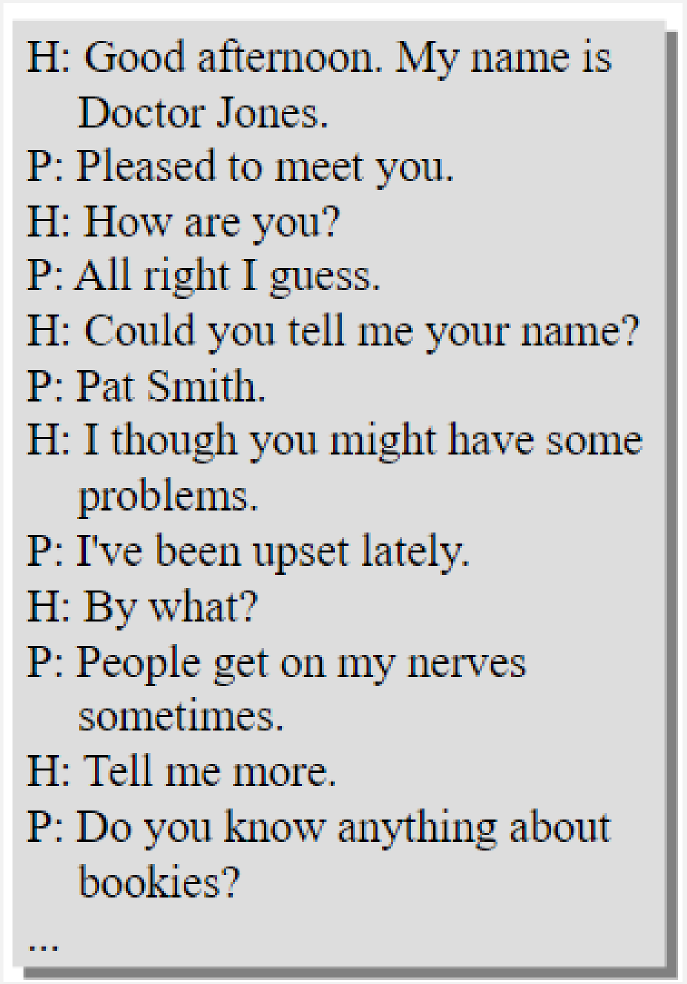
```
s/.* YOU ARE (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \
s/.* YOU ARE (depressed|sad) .*/WHY DO YOU THINK YOU ARE \1/
s/.* all .*/IN WHAT WAY/
s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE/
```

Step 3: use scores to rank possible transformations

Eliza style regular expressions

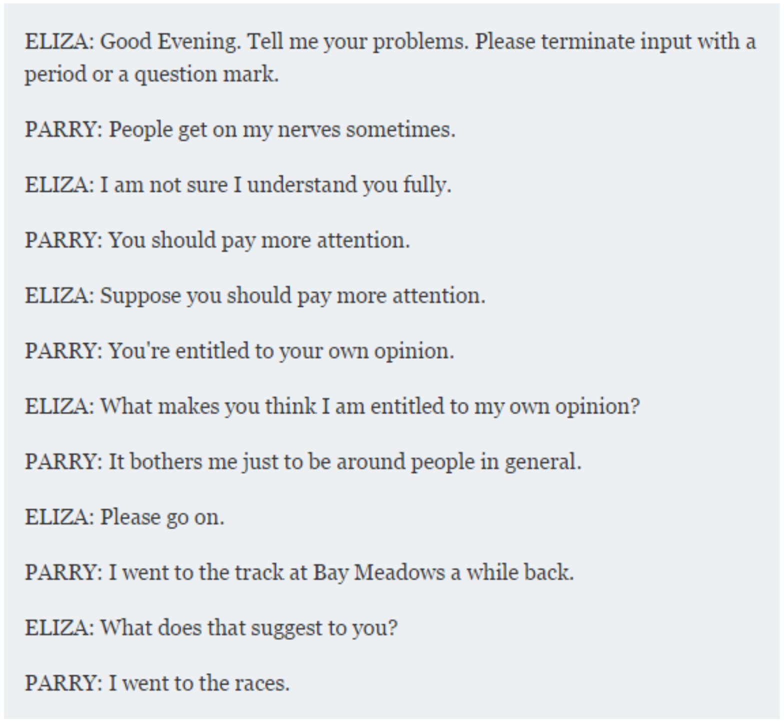
And then …

- Kenneth Colby developed PARRY in 1972
- PARRY
  - A computer Model of Paranoia developed for teaching psychology students
  - Simulated a paranoid schizophrenic
  - The first program to pass the Turing Test
  - Ignited a debate in the community on what was machine intelligence



H: Good afternoon. My name is Doctor Jones.
P: Pleased to meet you.
H: How are you?
P: All right I guess.
H: Could you tell me your name?
P: Pat Smith.
H: I though you might have some problems.
P: I've been upset lately.
H: By what?
P: People get on my nerves sometimes.
H: Tell me more.
P: Do you know anything about bookies?
...

Conversation between ELIZA and PARRY



ELIZA: Good Evening. Tell me your problems. Please terminate input with a period or a question mark.

PARRY: People get on my nerves sometimes.

ELIZA: I am not sure I understand you fully.

PARRY: You should pay more attention.

ELIZA: Suppose you should pay more attention.

PARRY: You're entitled to your own opinion.

ELIZA: What makes you think I am entitled to my own opinion?

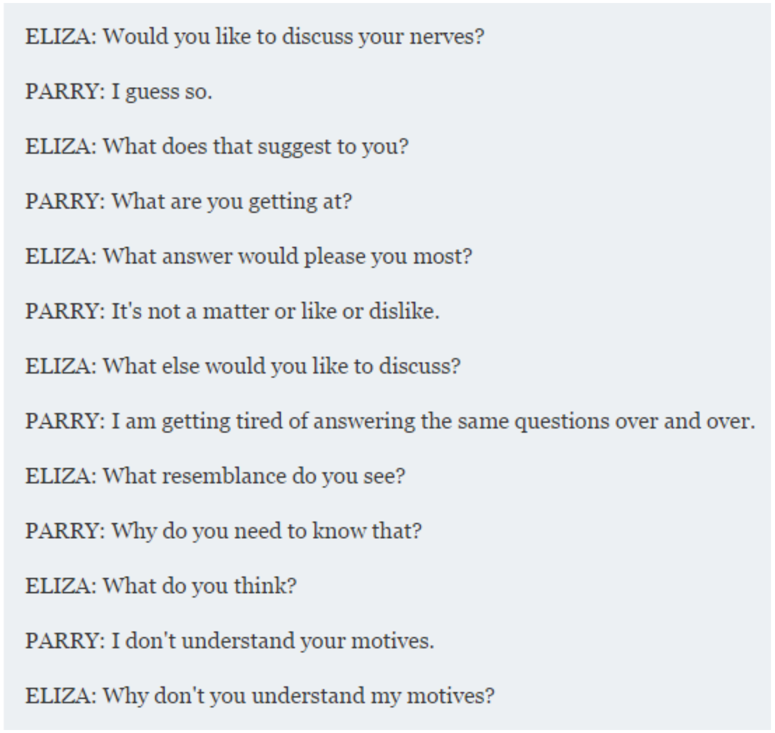
PARRY: It bothers me just to be around people in general.

ELIZA: Please go on.

PARRY: I went to the track at Bay Meadows a while back.

ELIZA: What does that suggest to you?

PARRY: I went to the races.



ELIZA: Would you like to discuss your nerves?

PARRY: I guess so.

ELIZA: What does that suggest to you?

PARRY: What are you getting at?

ELIZA: What answer would please you most?

PARRY: It's not a matter or like or dislike.

ELIZA: What else would you like to discuss?

PARRY: I am getting tired of answering the same questions over and over.

ELIZA: What resemblance do you see?

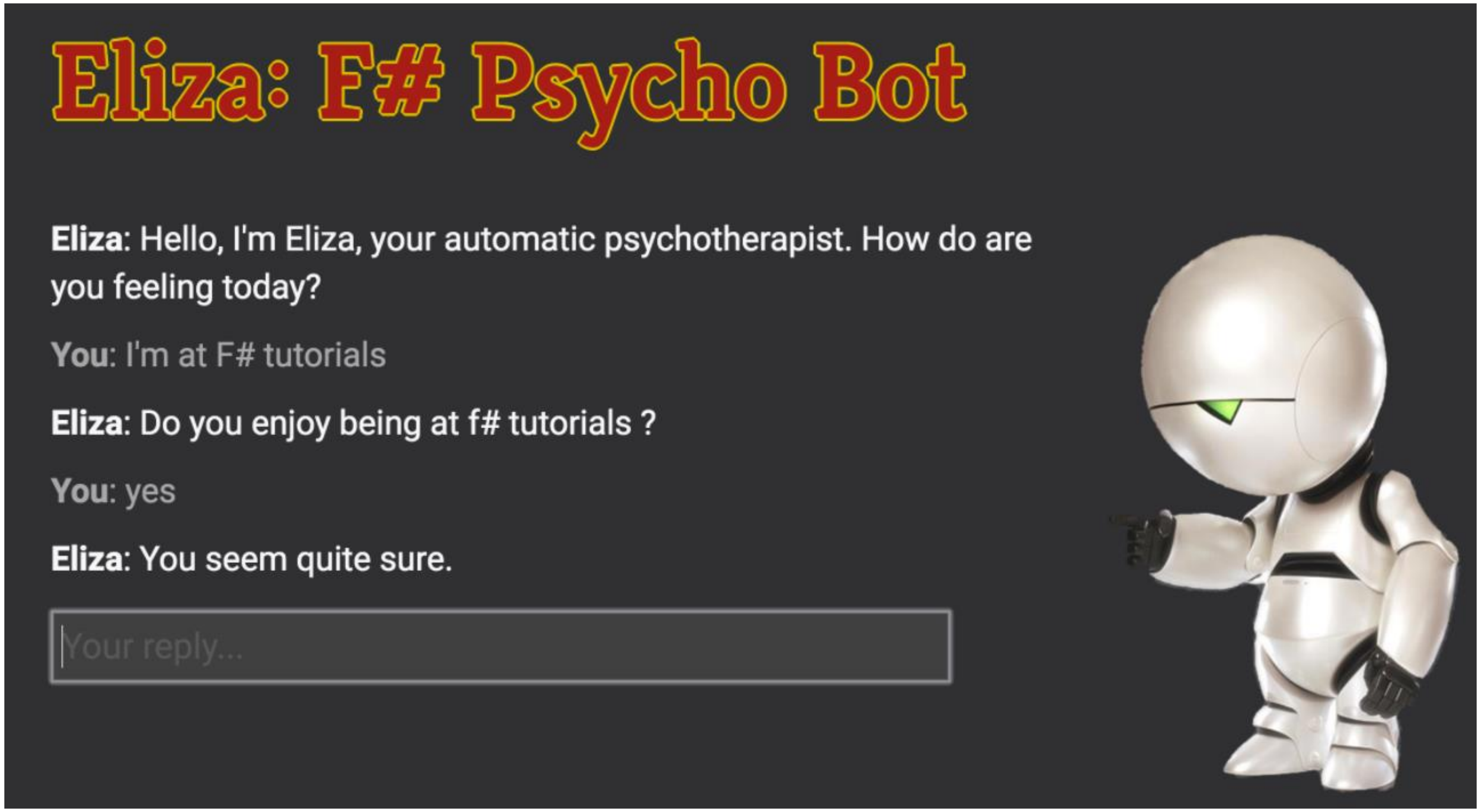
PARRY: Why do you need to know that?

ELIZA: What do you think?

PARRY: I don't understand your motives.

ELIZA: Why don't you understand my motives?

Programming Assignment



Eliza: F# Psycho Bot

Eliza: Hello, I'm Eliza, your automatic psychotherapist. How do are you feeling today?

You: I'm at F# tutorials

Eliza: Do you enjoy being at f# tutorials ?

You: yes

Eliza: You seem quite sure.