Chatbots

Tipos
- Conversacional
- Orientado a Tarefas

Principais Dificuldades
- Contexto
- Coerência
- Pragmática

Arquitetura de Referência
- Natural Language Understandig
- Gerenciador de Diálogos
- Conhecimento do Domínio
- Natural Language Generator

Projeto Baseado em Regras
- Estados
- Frames
- Inferência

Projeto Baseado em Corpus
- Inference Retriving
- Machine Learning

Network Working Group V. Cerf
Request for Comments: 439 SU-ERL
NIC: 13771 21 January 1973

PARRY Encounters the DOCTOR

The session recorded below actually happened on 18 September 1972 and is herewith reproduced for your entertainment.

Good Evening.

Tell me your problems. Please terminate input with a period or a question mark.

*People get on my nerves sometimes.

I am not sure I understand you fully.

*You should pay more attention.

Suppose you should pay more attention.

*You're entitled to your own opinion.

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

*It bothers me just to be around people in general.

Please go on.

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

What does that suggest to you?

*I went to the races.

What else would you like to discuss?

*Horseracing interests me.

Why do you say that just now?

*There's a lot of crooked gambling going on there.

I am not sure I understand you fully,

Módulo *nltk.chat*

A class for simple chatbots. These perform simple pattern matching on sentences typed by users, and respond with automatically generated sentences.

- nltk.chat
  - nltk.chat.eliza
  - nltk.chat.iesha
  - nltk.chat.rude
  - nltk.chat.suntsu
  - nltk.chat.util
  - nltk.chat.zen

nltk.chat.util module

class `nltk.chat.util.`**`Chat`**(*pairs*, *reflections={}*)

Bases: `object`

`converse`(*quit='quit'*)

`respond`(*str*)

Generate a response to the user input.

Parameters: str (*str*) – The string to be mapped
Return type: str

```
import re
import random

class Chat(object):
    def __init__(self, pairs, reflections=
    def _compile_reflections(self):
    def _substitute(self, str):
    def _wildcards(self, response, match):
    def respond(self, str):
    def converse(self, quit="quit"):
```

CATEGORIES, contained in the <category></category> tags, define a rule or unit of knowledge in AIML.

WILDCARDS like * allow keyword matching, and in this example would match any phrase beginning with "Hi".

```
<category>
  <pattern>HI *</pattern>
  <template>Hello world!</template>
</category>
```

PATTERNS, written in CAPS with no punctuation within the <pattern></pattern> tags, match the client's **input** to the bot.

TEMPLATES, contained in the <template></template> tags, define a bot's **output** returned when an input is matched.

```
from __future__ import print_function
from nltk.chat.util import Chat, reflections

myPairs = (

  (<REGEX>,
  ( STR_0,
    STR_1,
    STR_n)),

    ...
)

myBot = Chat(myPairs, reflections)
myBot.converse(quit={STR|REGEX})
```