Instituto Tecnológico de Estudios Superiores de Monterrey

Programming Languages TC2006

Final Project

Prolog Chatbot

Benjamín Valdés AguirrePhD

Karla Daniela Romero Pérez

A01704384

Table of contents

Context of the problem

It is known that communication is hard to achieve in big companies, especially when teams are spread all across the world. Issues like not speaking the same language, time zones and general misunderstandings arise when a process is trying to be communicated.

Sometimes a single email could be enough when you want to share how to do something to a new team, but many times they go unnoticed and the users have to check it consistently before they memorize the process.

Another alternative often used by organizations is creating a FAQ page somewhere in the web, however that can lead to security issues and revealing information to teams that shouldn't be seeing it.

One of the main places where I have seen this problem for the first time is in the company where I am currently working, a global organization that makes food for many countries who recently got a new office in Brazil. The communication between teams has been an issue because the new employees are having a hard time getting used to the new processes and the regional teams are very busy aiding them when they need it.

My proposal is to use chatbots. In this section I will explain some of the reasons why I suggest this.

The origin of chatbots

Since the creation of computers, it has been the aim of computer scientists to simulate the human mind with a machine. One of the first approaches to this was chatbots, an Artificial Intelligence (AI) program that chats with humans, either through chat or through voice, like Amazon's Alexa.

The first chatbot was called ELIZA, developed by Joseph Weizenbaum in the 1960’s, even though it did not use AI, its workflow is based on pattern matching and substitution. The engine would parse through the words entered by users and pair them to a group of possible responses. ELIZA was the beginning of natural language processing.

After this, many other chatbots have emerged, like Cortana, Google Assistant, etcetera.

The success of chatbots

According to a survey done by AI Multiple to chatbot vendors, agencies and consultancies, there are a huge amount of benefits to chatbots, specially for users, these are listed below (Top 12 Benefits of Chatbots in 2021: The Ultimate Guide, 2018):

Available 24/7

Chatbots are one of the best ways to manage a big amount of users requesting the same information at the same time. Instead of having to wait for an email reply or for someone to aid you, you can just ask a bot and get an instant reply, for as many users as the software is capable of managing at the same time.

Recorded answers

Even though you can save the information given in an email, sometimes it is hard to search in the huge amount of emails for a specific one. In this case chatbots provide the possibility for the users to take screenshots or even the chatbot can return a summarized version of the conversation.

Patience

No matter how many times you ask a bot the same question, it will answer again and again, whereas in a mail, you can't ask much if you have any doubts.

Better user experience

Chatbots can deliver more of a personalized experience comparing it to an email or an informative web page. Users tend to feel more comfortable asking a computer the stuff they don't understand, as well as it feels more natural.

Solution

To aid this problem, I created a chatbot using Logic Programming with Prolog, which registers a few mandatory data and gives information about the process of adding a printer to a global and local network in order to integrate it to SAP. This particular situation is addressed because it is needed in the company where I am doing my professional practices but in general, you can add any process you want to share with your organization.

The rules of this chatbot work in a way that when you are done with the mandatory information you can chat with it in a more casual way and ask about its name, and some other stuff. When the user is ready to leave, they can say goodbye and the chatbot will return a report of the important information of the conversation.

This solution works similarly like ELIZA, in the sense that it uses Natural Language Processing along with Definite Clause Grammar, a way of expressing grammar in logic programming. Both of these terms will be explained in a moment.

Currently, the chatbot is working in the console line because the way it operates is by write() commands. However it is possible to add a graphic interface transforming the write() commands to lists and keeping them as facts so that they print on the GUI.

Why Prolog?: Natural Language Understanding

Prolog is a language whose main features include unification and backtracking, which make it very powerful for Natural Language Processing, this can be defined as a way of processing human languages with a computer or artificial intelligence (8 Natural Language Processing in Prolog, n.d.).

NLP is composed of 6 steps that allow the comprehension of human language, which are explained below (edureka, 2018).

1. **Tokenization**

   The process of breaking strings into smaller structures that allow the parsing. In this solution we break it into words, as elements of a list, taking the ASCII values that are special, such as ? or !.

```
%% read_in will process the natural language of the input of the user calling two other functions

%% M is the entered message [hello]
%% L is the list of ASCII values for each of those characters [72,101,108,108,111,10]
read_in(M):-initread(L),words(M,L,[]).

%% initread will parse through the entered message list to translate to the ASCII values
initread([K1,K2|U]):-get_code(K1),get_code(K2),readrest(K2,U).


%% readrest checks for the different values of a list of ASCII codes, ignoring question marks and exclamation marks
readrest(63,[]):-!. %% Question mark
readrest(33,[]):-!. %% Exclamation mark
readrest(10,[]):-!. %% Line feed

%% Skip the character if it is a control one
readrest(K,[K1|U]):-K=<32,!,get_code(K1),readrest(K1,U).
%% Else, get its code and add it to the list
readrest(_K1,[K2|U]):-get_code(K2),readrest(K2,U).
```

Code implementation of tokenization

2. **Stemming**

   Is normalizing the words into its root form, for example

Affectation, Affected, Affection, Affective = Affect

Usually this will be taken from the first letters of all the words, however it can usually be wrong

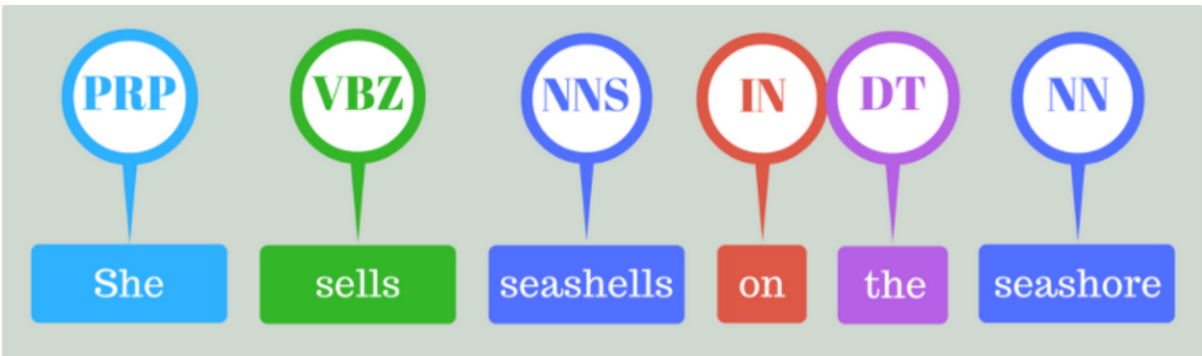
3. **Lemmatization**

Similar to stemming, but it takes into consideration the morphology of words, therefore it is necessary to have a detailed dictionary of the roots of the words (lemma). One case where lemmatization could be better is

Gone, Went, Going = Go

4. **POS Tags**

POS stands for Paths Of Speech which is the parts of sentences, for example, determiners, nouns, verbs, pronouns, adverbs, prepositions, etcetera. It indicates how words work within a sentence.



Example of a sentence using POS tags

In my solution we do this by using Definite Clause Grammar, which will be explained further in this document.

5. **Named Entity Recognition**

During this step, our software should recognize names of movies, organizations and locations that are relevant to the context where the chatbot will be used so that it doesn't process it as unknown.

In the code we do this by defining a knowledge base of possible names as well as printer brands. This helps also when we want to verify that the user entered a valid input.

```
nameList('karla').
nameList('cris').
nameList('jucemar').
nameList('alysson').
```

Named Entity Recognition in Prolog

6. Chunking

The process of grouping words or tokens again in order to process them to get insights and information from the user's inputs. For example:

Determiner + Adjective + Noun = Noun Phrase

The yellow car

In our code we can see this implemented also with Definite Clause Grammar syntax where we group the tokens we defined earlier

```
object_phrase(op(X,Y)) --> noun_phrase(X), adverb(Y).
object_phrase(op(X, Y)) --> object_pronoun(X), adverb(Y).

noun_phrase(np(X, Y)) --> determiner(X), noun(Y).
noun_phrase(np(Y)) --> noun(Y).
```

Chunking in Prolog

Like I said before, Prolog's nature allows us to represent all this easily, and along with Definite Clause Grammar I was able to form a working chatbot.

Definite Clause Grammar

Definite Clause Grammar is a way to express human language grammar in logic programming as a set of definite clauses. They are able to perform recognition and parsing using top-down depth-first search, specifically in Prolog. One could say that DCG is a way of translating grammar rules into prolog clauses and predicates.

DCG Syntax

Rules

A DCG sentence is composed of many DCG rules, which at the same time have two main parts:

head: non terminal symbol

body: a list of elements separated by comma, that can be a non terminal symbol, a list of constants (found between [ ]) or prolog predicates (found between { }).

A DCG rule in prolog looks like this:

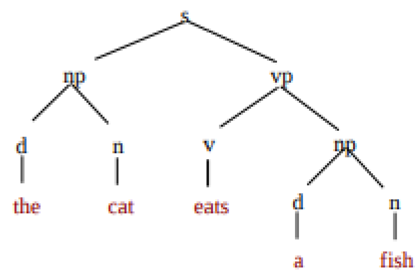
```
% a word is built either with alphanum or only digits
word(U1) --> [K],{lower_case(K,K1)},!,alphanums(U2),{name(U1,[K1|U2])}.
```

An example of a DCG rule in Prolog

In this example, I am defining what a word is, which can be a combination of alphanumeric characters. Note in this example that:

- [K] consumes a character from the word and unifies it with the variable K
- {lc(K,K1)} is a call to a Prolog predicate that turns letters to lowercase
- alphanums(U2) calls another DCG using attributes, often called attribute grammars. Using attributes along with Definite Clause Grammar rules helps to transport tokens across the grammar rules.

A graphic representation of DCG can help explain better (Rigau & Claramunt, n.d.):



A parsing tree for a sentence (Rigau, 2018).

Here we can see that a **sentence** is formed by a **noun phrase** and a **verb phrase**.

In code, it would look like this:

```
sentence(s(NP,VP)) --> noun_phrase(NP), verb_phrase(VP).
```

A **noun phrase** at the same time is composed by a **determiner and a noun**

```
noun_phrase(np(D,N)) --> det(D), noun(D).
```

And a **verb phrase** has a **verb and another noun phrase**

```
verb_phrase(vp(V,NP)) --> verb(V), noun_phrase(NP).
```

The rest of rules are **dictionaries** and have the possible tokens for each, for example, the determiners can be **the, a, an, etcetera.**

```
det(d(the)) --> [the].

det(d(a)) --> [a].
```

In the end, like it was mentioned before, Prolog has a built-in "engine" that allows us to write grammar rules in the form of DCG and it automatically translates them to predicates, making it easier to parse and understand human language.

All of the concepts explained in this section are the core of the solution I created, and I consider that understanding this, it is very easy to create a chatbot for any context, and any purpose.

Results and tests

The chatbot is currently uploaded in this github repository with a GNU licence, and works as follows:

- When running, it shows a welcome message and the prompt for the user to write anything

```
PS C:\Users\kdani\OneDrive\Escritorio\KelloggChatbot> swipl --quiet bot_logic.pl
Alexa: Hi there!
Alexa: You can ask me about the processes to add a new device to our global network.
Alexa: How can I help you today?
You: 
```

- After saying Hello, the chatbot asks for the user's name and as long as it isn't a valid one, you won't be able to proceed to any other conversation

```
PS C:\Users\kdani\OneDrive\Escritorio\KelloggChatbot> swipl --quiet bot_logic.pl
Alexa: Hi there!
Alexa: You can ask me about the processes to add a new device to our global network.
Alexa: How can I help you today?
You: Hello, I need help
Alexa: What's your name?
You: No
Alexa: I need to know your name!
You: I don't know
Alexa: That's not your real name...
You: Karla
Alexa: Nice to meet you. What brand and model is the printer?
You: 
```

- The same happens when the bot asks for the printer brand

```
Alexa: Nice to meet you. What brand and model is the printer?
You: Fujifilm
Alexa: That's not a real printer...
You: I don't know the brand
Alexa: Are you sure?
You: Yes
Alexa: That's not a real printer...
You: Bye
Alexa: That's not a real printer...
You: Xerox
Alexa: Right now, what's the inside IP for the device?
You:
```

- The next step is having the private IP for the printer. This response does not have a validation since the amount of valid IPs can be very extensive; however, this makes the user have the IP address in hand to proceed with the next step
- After introducing the IP, the chatbot only gives information ignoring the user's input. When it ends, the user is free to ask and play around with the bot.

```
Alexa: Right now, what's the inside IP for the device?
You: 192.168.1.3
Alexa: Okay, please assign a Company Network IP and document it
You: Ok
Alexa: The next step is to ask the regional team to add the translation to the fortigate
You: Done
Alexa: Ask the Security global team to add this IP to the firewalls
You: Alright
Alexa: To request a printing queue, contact the Basis Team
You: Ok
Alexa: Finally, run the proper tests.
You: Great!
Alexa: Awesome
You:
```

- In the free mode, the bot is programmed to answer basic questions such as "What's your name?", "Are you ok?", and even if the user enters nonsense, the bot has some random questions to keep a "logical" conversation

```
Alexa: i am awesome !
You: whats your name?
Alexa: I'm Alexa!
You: are you ok?
Alexa: i am awesome !
```

```
Alexa: i am awesome !
You: a
Alexa: So, what are your thoughts to make me better?
You: None
Alexa: Ok, thanks. Have I been helpful?
You:
a
Alexa: Hmm. Do you think that I am a human?
You: a
Alexa: Okay. Did you find any of the talks interesting?
You: a
Alexa: Great!
```

- If the user enters many nonsense words or phrases, the bot will stop responding, but when the user types "Bye", it will say goodbye and print the report of the conversation with the main and mandatory information.

```
Alexa: Great!
You: a
You: no
You: Bye
Alexa: Have a nice day!

--- Conversation history ---
Your name:  karla

[So, what are your thoughts to make me better?] : none
[Ok, thanks. Have I been helpful?] : a
[Hmm. Do you think that I am a human?] : a
[Okay. Did you find any of the talks interesting?] : a

[What's your name?] : no
[Nice to meet you. What brand and model is the printer?] : fujifilm
[Right now, what's the inside IP for the device?] : num(192) . num(168) . num(1) . num(3)
[Okay, please assign a Company Network IP and document it] : ok
[The next step is to ask the regional team to add the translation to the fortigate] : done
[Ask the Security global team to add this IP to the firewalls] : alright
[To request a printing queue, contact the Basis Team ] : ok
[Finally, run the proper tests.] : great !
```

Conclusions

To summarize, a chatbot is a combination of Natural Language Processing and Definite Clause Grammar, that allow us to make a computer understand and analyze information in the form of natural human language.

The implementation of these concepts was essential to achieve this project since it constitutes the core of it. As we have seen before, this can be implemented in other programming languages, however it has been proven that Prolog is one of the most powerful for this kind of task. Some of the reasons are the built-in DCG syntax, unification and backtracking feature it has.

In my experience using and testing this software I was able to see some real improvements in the communication among teams; regional managers could just deploy this chatbot and save time by letting the software explain the processes to users as many times as necessary.

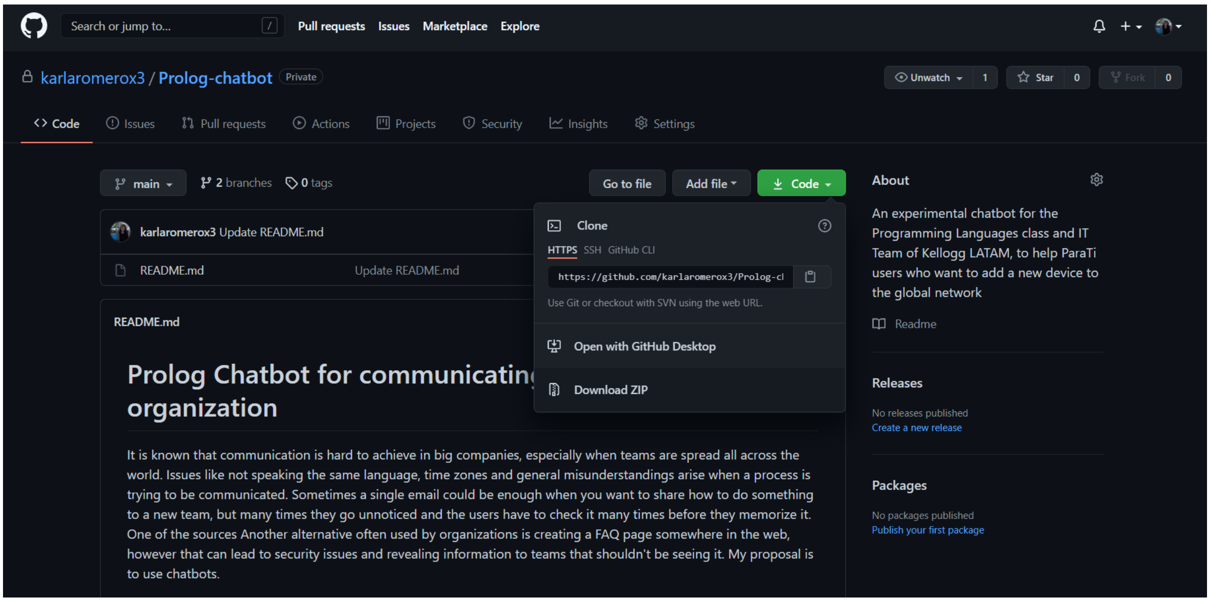
Some improvements that can be done in the future are the creation of a graphic interface (I suggest a web-based application) and implementing more artificial intelligence technologies in order to ensure the chatbot has a better understanding of the human language.

Setup instructions

Please follow these instructions to run this project on your computer.

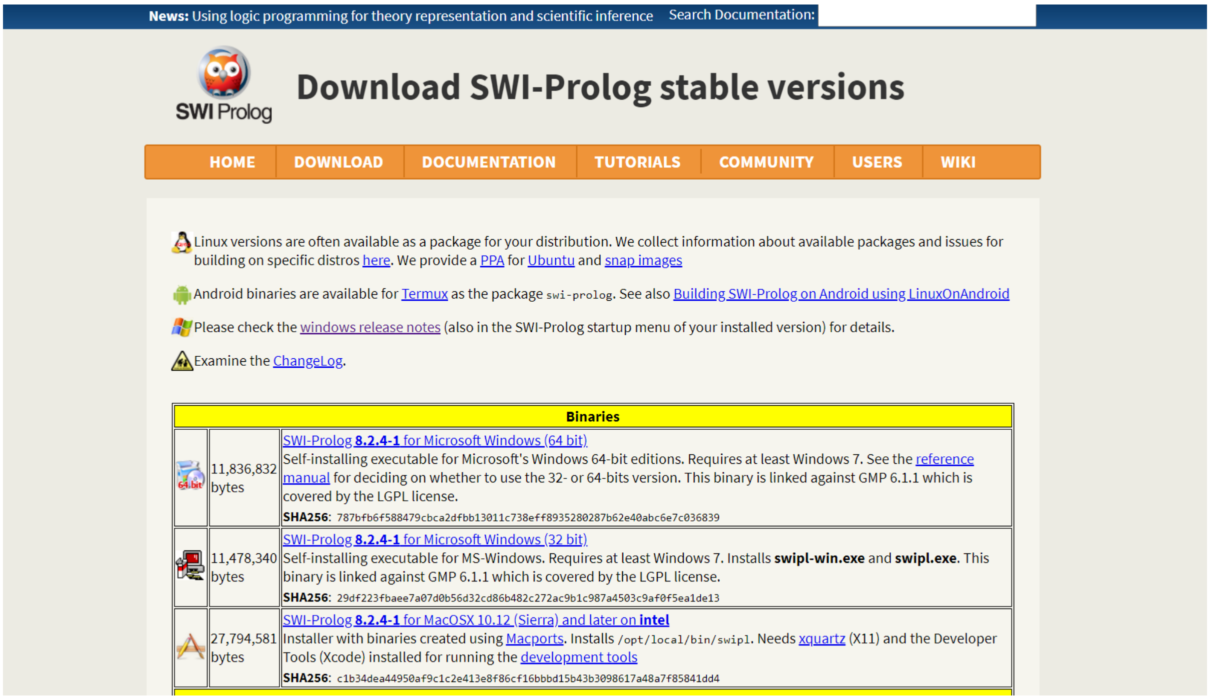
1. Clone the github repository

Open the github repository and download it as a ZIP file



2. Install SWI-Prolog

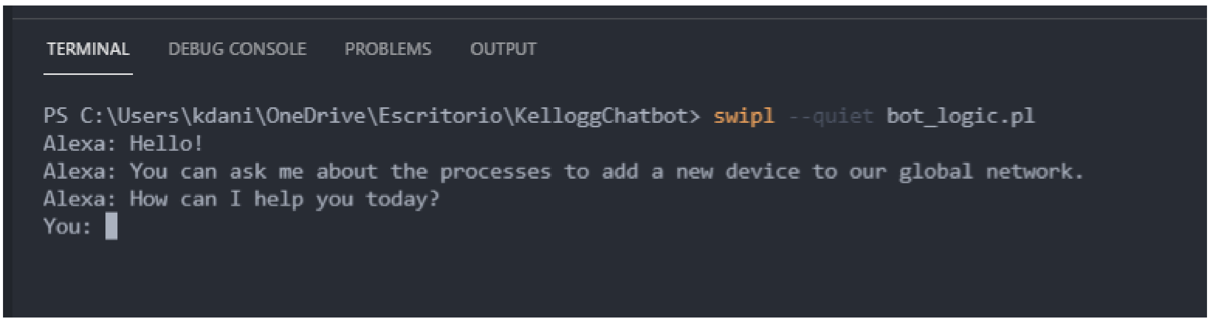
Enter SWI-Prolog downloads and download the version of the Prolog compiler that fits your computer



3. Run the code

Unzip the file you downloaded from github and access the folder where the .pl file is located.

Then run the command **swipl --quiet bot_logic.pl**



References

Bitter, C., Elizondo, D. A., & Yang, Y. (2010). Natural language processing: a prolog perspective. ARTIFICIAL INTELLIGENCE REVIEW, 1–2, 151.

8 Natural Language Processing in Prolog. (n.d.). Retrieved June 3, 2020, from https://www.cs.unm.edu/~luger/ai-final2/CH8_Natural%20Language%20Processing%20in%20Prolog.pdf

D'Souza, J. (2018, April 4). *Learning POS Tagging & Chunking in NLP*. Medium; GreyAtom. https://medium.com/greyatom/learning-pos-tagging-chunking-in-nlp-85f7f811a8cb

edureka. (2018). Natural Language Processing In 10 Minutes | NLP Tutorial For Beginners | NLP Training | Edureka [YouTube Video]. On *YouTube*. https://www.youtube.com/watch?v=5ctbvkAMQO4

Ina. (2017, October 12). *The History of Chatbots - From ELIZA to Alexa*. Chatbot and Voice Assistant Solutions from Onlim. https://onlim.com/en/the-history-of-chatbots/#:~:text=The%20first%20chatbot%20ever%20was

Main Page. (2021, February 3). Wikipedia. https://en.wikipedia.org/?url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FDefinite_clause_grammar

Rigau, G., & Claramunt. (n.d.). *Prolog & NLP*. Retrieved May 31, 2021, from https://adimen.si.ehu.es/~rigau/research/Doctorat/LSKBs/16-Prolog.pdf

Top 12 Benefits of Chatbots in 2021: The Ultimate Guide. (2018, March 29). Research.aimultiple.com. https://research.aimultiple.com/chatbot-benefits/#:~:text=Increased%20customer%20interaction