INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES
DE MONTERREY

DEPARTAMENTO DE CIENCIAS E INGENIERÍA

Selecting Movies with Chatbots

by

Adriana Paola Salinas García

Programming Languages Final Project

QUERÉTARO, QRO. DECEMBER 2021

ABSTRACT

Chatbots are software applications that use artificial intelligence and natural language processing that simulate the way a human being behaves to understand what a live human wants, either via voice or text, and guides them to their desired outcome. This project has the purpose of showing one practical use of this technology on a common issue nowadays in a predicate-based programming paradigm. During the chapters we will describe the problem presented, explain its solution implementation on the selected paradigm, and finally show its results and statistical analysis. I will also present the code structure to apply the solution in other contexts.

Contents

Chapter 1

Context of the problem

1.1 Netflix and the Paradox of Choice

Streaming services sell themselves as platforms of media content that grow over time as new movies and series are added to their libraries in order to give more power of choice to the consumer. However, what happens when the choice selection is too big that we can't decide what to watch? In a world with almost infinite options, we risk missing the simple pleasures by constantly searching for something better, and therefore we end up profoundly disappointed with our choices. [1]

This phenomenon is described by the psychologist Barry Schwartz as the Paradox of Choice. Basically, when we are overwhelmed with choice, we end up on a paralysis analysis in which we choose nothing at all. [1][2] With streaming services, users now spend more time deciding what media content to see (usually more than 10 minutes) than actually watching them, and in the end they prefer to stick to the movies that already know rather than trying new ones. And this issue is exponentially increasing as more content becomes available in these platforms.

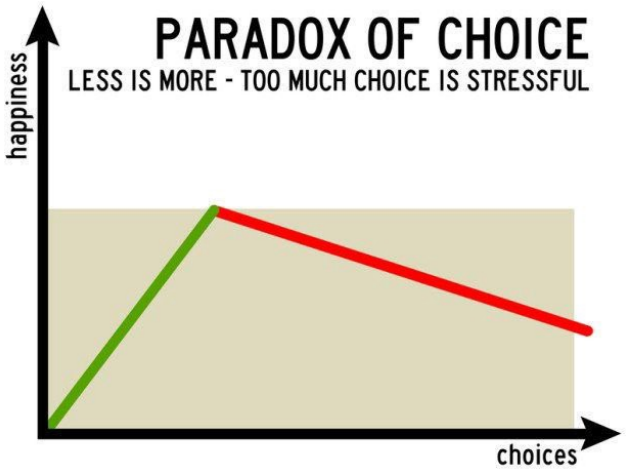

Figure 1.1: The Paradox of Choice graph

In 2000, psychologists Sheena Iyengar and Mark Lepper from Columbia and Stanford University published a study in which they proved the Paradox of Choice by asking the test subjects to pick a jam jar from a table with a variety of them. During their research, they found that when presented with 24 choices, only 3% if people went to purchase a jar, while with only 6 choices, the percentage rose up to 33%. The study showed that, while choice seems appealing at first sight, choice overload generates the wrong results as it paralyzes the consumer. This is because, when we spend time and effort choosing between various alternatives, we expect our extra efforts to pay-off, leading to us having higher expectations from the chosen product, which may disappoint us more than expected. [1][2]

1.2 Why chatbots?

Chatbots are usually used by companies to engage with their customers by serving relevant resources or services according to their demands in faster times than classic customer services. The more information is feeded to a chatbot, the more precise and tailored a solution will be given to the user, so companies have been investing in AI technologies to improve their quality of understanding and decision making. In this case, it would help the user to narrow down their movie choices on any given moment when using Netflix or any other streaming service.

Chapter 2

State of the art

2.1 What is a chatbot?

A chatbot is defined as a computer program that simulates and processes human conversation, allowing humans to interact with digital devices as if they were communicating with a real person. They can be as simple as rudimentary programs that answer a simple query with single-line responses, or as sophisticated as digital assistants that learn and evolve to deliver increasing levels of customized service as they gather and process information.

The main purpose of chatbots is to satisfy business needs in business-to-consumer (B2C) or business-to-business (B2B) environments by delivering relevant information and completing tasks faster than classic services, therefore reducing costs and improving customer experience. However, they are not intended to replace human agents, especially for more complex problems or scenarios. [3]

2.2 History of chatbots

The history of chatbots dates back to the invention of the computer with Alan Turing. The Turing test was proposed as a criterion of intelligence for a program to impersonate a human in a real-time written conversation. In 1966, MIT professor Joseph Weizenbaum developed the first chatbot named ELIZA which was able to emulate a human conversation throught the use of clue keywords and pre-programmed responses. Then, in 1972, psychiatrist Kenneth Colby constructed PARRY as a chatbot able to imitate a patient of schizophrenia to simulate the disease.

With the arrival of the Internet, in 1995, Richard Wallace build ALICE, a universal language processing chatbot that uses heuristic pattern matching to simulate an online conversation with a women. Finally, the research of natural language processing led to the development of various chatbots by big tech companies. Today, the market is dominated by chatbots like Siri, Google Now, Cortana and Alexa, all able to perform actions through voice requests based on the contextual information of the user. [4]

Figure 2.1: Chatbot example

2.3 Types of chatbots

2.3.1 Task-oriented/Declarative/Guidelines

Declarative chatbots are single-purpose programs that only perform one function. They generate automated but conversational responses to user inquiries. They can handle common questions, so their capabilities are very basic and limited to their programming. Their highly specific and structured interaction level make them very useful on support and service functions.

2.3.2 Data-driven/Conversational/Machine Learning

Data-driven or predictive chatbots are more sophisticated, interactive and personalized programs that use full natural language processing and machine learning to apply predictive intelligence and analytics to enable custom functions based on user profiles and past behavior. Often called virtual assistants, they are able to provide recommendations, anticipate needs or even start conversations. [4][5]

2.4 Uses

In businesses, repetitive or manual tasks slow agent productivity and frustrate customers. Automation with chatbots speeds things up. They allow companies to easily resolve many types of customer queries and issues while reducing the need for human interaction. [4]

1. Call Center support: They allow users to help themselves without the need of an agent. They complete tasks such as changing a password, requesting a document or scheduling an appointment.

2. Enterprise support: They help employees to manage business information, such as contact information, inventory management or customer relationship management (CRM).

3. Digital assistants: They help consumers with their day-to-day lives and do activities at distance, such as ordering groceries, booking dates or managing IoT devices.

2.5 Benefits

While chatbots can't replace humans, they help speed up the customer support experience by answering easy questions and collecting information to solve a case quickly. For example, in customer service, chatbots help in the following ways. [4]

1. Reduce customer wait time: Chatbots reduce the time customers spend waiting in line as get immediate answers to common questions in a chat window.

2. Resolve support cases. Chatbots resolve support cases fast by answering straightforward questions for customers to make them happier effortlessly, therefore reducing the number of cases to human agents.

3. Serve up resources for customers: Chatbots can instantly welcome customers and efficiently direct them to the resources they seek.

4. Identify leads for the business: By handling initial support interactions with a customer, chatbots help open conversations for service agents to follow up and personalize future customer interactions.

Chapter 3

Solution

3.1 Programming paradigm

3.1.1 Logic programming

Logic programming is a programming paradigm where statements express facts and rules about a problem within a system of formal logic. A program is run by presenting some query and seeing if this can be proved against those known rules and facts.

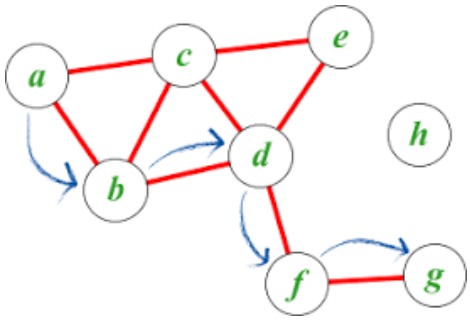

Figure 3.1: Logic programming idea

3.2 Prolog

3.2.1 Prolog structure

Prolog is a logic programming language that uses predicates and first-order logic suited to programs that involve symbolic or non-numeric computation. For this reason it is a frequently used language in artificial intelligence and natural language processing where manipulation of symbols and inference about them is a common task such as searching databases, voice control systems, and filling templates [7].

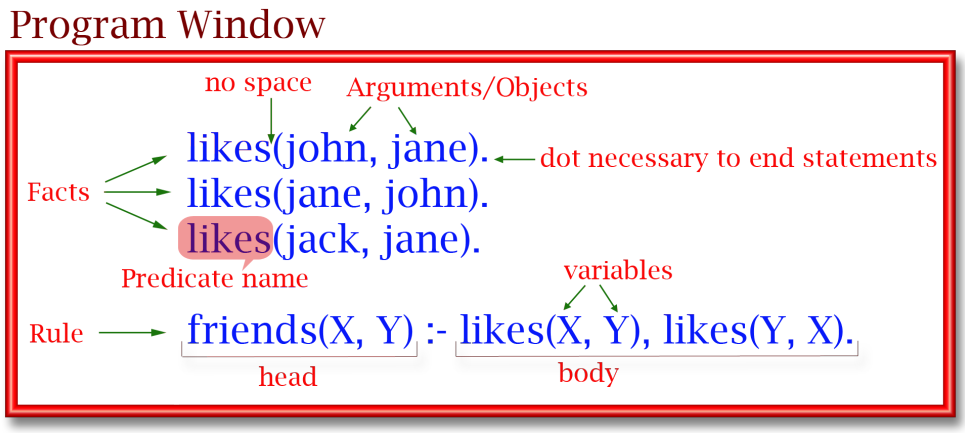

Figure 3.2: Prolog code example

3.2.2 Prolog implementation

My chatbot implementation in Prolog starts with the definition of the facts. Since the purpose of the project is to help users of streaming services such as Netflix to select a movie faster, we must define the characteristics of the movies the chatbot is going to recommend as facts. As shown in Figure 3.3, I define the characteristics of the movies under the format **movie-attribute(attribute, movie-name)**.

```
33 movie_genre(fiction, axl).
34 movie_genre(fiction, dune).
35
36 movie_moment(night, apostle).
37 movie_moment(afternoon, host).
38 movie_moment(morning, cadaver).
39 movie_moment(afternoon, oxygen).
```

Figure 3.3: Code facts

Then, how are we going to tell the user which movie to watch based on his preferences? Well, I defined a counter for each movie in which we add one to those that have the attribute selected by the user. In other words, if the user for example wants to see a horror movie, we add one to the counter of all movies which are classified by our facts as part of the horror genre. By the end of the questionnaire, the chatbot will output the movie whose counter value is the highest. In case two or more movies have the same value, the chatbot will default to the first one written on the facts.

After stating all of the movies facts, I proceeded with the program introduction. The chatbot will firstly greet you by explaining both the context of the problem and purpose of the questionnaire. After this, it will start asking you questions about your preferences to recommend you a movie.

```
counter(apostle, 0).
counter(host, 0).
counter(cadaver, 0).
counter(oxygen, 0).
counter(us, 0).
counter(aftermath, 0).
counter(eli, 0).
counter(run, 0).
counter(sputnik, 0).
counter(platform, 0).
```

Figure 3.4: Movie counter

```
begin():-
    write('It has happened to all of us that, regardless of the variety of movies
           or series that are on Netflix, choosing one is always difficult and
           can take us much more than we think.
           Not only that, after making the difficult decision we are waiting to
           see if we like or not what we have chosen.
           So, when it comes to choosing what to see,
           several things have to be taken into account,
           for how long we are willing to invest,
           the tastes we have in terms of gender and even our state of mind.
           If you have found yourself in that situation of wasting more than 40 min
           selecting something to entertain yourself, then this tool is for you.'), nl,

    write('I invite you to answer the following questions that
           will help me determine the best option for you.'), nl,
    write('Please answer with lowercase letters and without spaces.'), nl, nl.
```

Figure 3.5: Chatbot instructions

The questions are structured through rules that ask you to type a keyword, which are read by Prolog. Then, those movies that contain the keyword in their facts as attributes, their counters will be updated. However, due to unification, we can't update the value from the counter facts directly, therefore we need another way to represent the addition of the counter. For this I created a new rule called **updateCont**, which retracts the counter facts from the matching movies and replaces them with the same one but with the previous counter value plus one.

Once the user has answered all the chatbot questions, the **contList** rule picks all the movies counter values and inserts them into a list. Finally, the **maxlist** rule extracts the biggest number from the list by iterating over it until reaching its end and storing the biggest number found so far. In the end, the chatbot will print the user the name of the movie it should see.

```
questionnaire(Z,N,L,R,Ans,Mov):-
    ask_genre(Z,N),
    ask_moment(Z,N),
    ask_partner(Z,N),
    ask_emotion(Z,N),
    ask_similar(Z,N),
    contlist(L),
    maxlist(L,0,R),
    counter(Ans,R),
    write('You should see the movie:\n'),
    upcase_atom(Ans,Mov),
    write(Mov).
```

Figure 3.6: Chatbot structure

```
updateCont(Z,N):-
    call(counter(Z,N)),
    retract(counter(Z,N)),
    NX is N + 1,
    assertz(counter(Z,NX)),
    call(counter(Z,NX)).

ask_genre(Z,N):-
    write('What is your favorite genre?'), nl,
    write('Horror'), nl,
    write('Romance'), nl,
    write('Fiction'),
    read(A),
    forall(movie_genre(A,Z), updateCont(Z,N)).
```

Figure 3.7: Question and updateCont rule

```
contlist(L):-
    findall(Z, counter(_,Z), L).

maxlist([],R,R).
maxlist([H|T],X,R):-
    H > X,
    maxlist(T,H,R).
maxlist([H|T],X,R):-
    H =< X,
    maxlist(T,X,R).
maxlist([H|T],R):-
    maxlist(T,H,R).
```

Figure 3.8: List processing

Chapter 4

Results

Our results proved that the chatbot reduced the time to select a movie dramatically. As shown in the Figure 4.2, on every single test, users where able to select a movie thanks to the chatbot recommendation in an average of 81.58% less time, reducing it down to less than five minutes. Also, Figure 4.1 shows that almost 2/3 of all users were satisfied with the movie recommended, proving the chatbot's level of effectiveness.

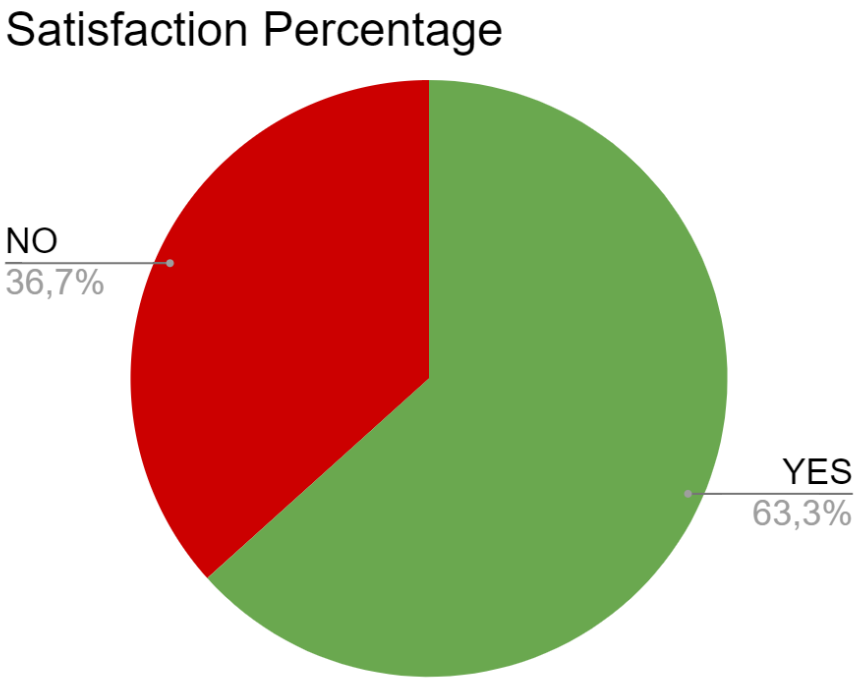

Figure 4.1: Test results

T	Actual Time (Minutes)	Charsi Time (Minutes)	Time saved (Percentage)	Satisfaction
1	32	3	90,63%	YES
2	21	3	85,71%	YES
3	15	2	86,67%	NO
4	19	3	84,21%	YES
5	35	3	91,43%	YES
6	30	2	93,33%	YES
7	24	2	91,67%	NO
8	12	4	66,67%	NO
9	8	3	62,50%	YES
10	5	3	40,00%	NO
11	13	2	84,62%	NO
12	24	3	87,50%	YES
13	34	3	91,18%	YES
14	18	3	83,33%	NO
15	36	3	91,67%	NO
16	22	3	86,36%	NO
17	17	3	82,35%	NO
18	20	3	85,00%	YES
19	29	3	89,66%	YES
20	18	4	77,78%	YES
21	37	4	89,19%	YES
22	6	4	33,33%	YES
23	9	4	55,56%	YES
24	42	4	90,48%	YES
25	44	4	90,91%	YES
26	21	4	80,95%	NO
27	35	5	85,71%	NO
28	30	3	90,00%	YES
29	12	1	91,67%	YES
30	16	2	87,50%	YES
		Optimization	**81,58%**	

Figure 4.2: Satisfaction results

Chapter 5

Conclusions

30 tests were done and the results are surprising. As previously seen on the results, all tests showed that the chatbot facilitated the movie selection of all users by a huge margin thanks to the preprogrammed answers inserted in it.

This proves that chatbots can be useful to help us with decision-making, and therefore streaming services should implement them in their platforms to increase watch time and customer satisfaction. If we added more questions to the chatbot, the recommendation list would be broader and more tailored to every user's preferences.

With the use of this program, users can say goodbye to the long times deciding which media to watch. I expect that the use of chatbots can be expanded towards other day-to-day decisions to stop thinking more than enjoying our lives. If we keep understaning them better, they will understand us better as well.

Chapter 6

Usage

6.1 Repo

The repository of the code is located here, it can be accessed by everyone and it includes the instructions to execute it.

6.2 Instructions

This repo also includes a .jar file to run the project without compiling it in Java. The steganography program lets you hide and recover a message from an image and prints the parallel and sequential execution times. The jar file needs some information:

- Copy and paste the code in the Online Prolog IDE
- Run the following code in the IDE command line: **charsi(Z,N,_L,_R,_Ans,_Mov)**

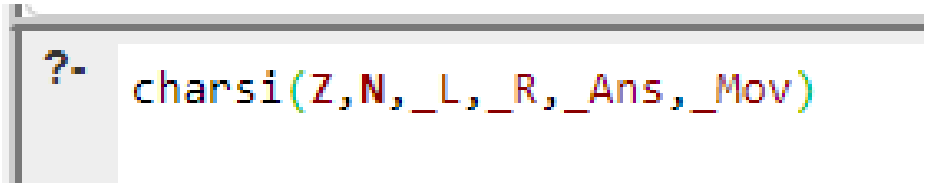

Figure 6.1: Online Prolog IDE Command Line

Bibliography

[1] Stark, J. (2019), How the paradox of choice keeps you scrolling through Netflix, *The Sydney Morning Herald*, Obtained from: https://www.smh.com.au/lifestyle/health-and-wellness/how-the-paradox-of-choice-keeps-you-scrolling-through-netflix-20190701-p5231b.html

[2] Gupta, S. (2020), Why is less more? -Choice Overload Phenomenon. *Medium*, Obtained from: https://medium.datadriveninvestor.com/why-is-less-more-choice-overload-phenomenon-29e43f19b316

[3] Ina (2017), The History Of Chatbots - From ELIZA to ALEXA, *Onlim*, Obtained from: https://onlim.com/en/the-history-of-chatbots/

[4] Higgins, A. (2021), What Is a Chatbot? How Is it Changing Customer Experience?, *SalesForce*, Obtained from: https://www.salesforce.com/blog/what-is-a-chatbot/

[5] Oracle (2021), What Is a Chatbot?, *Oracle*, Obtained from: https://www.oracle.com/ph/chatbots/what-is-a-chatbot/

[6] Frankenfield, J. (2020), Chatbot, *Investopedia*, Obtained from: https://www.investopedia.com/terms/c/chatbot.asp

[7] Treasure-Jones, T. (1996), Introduction to Prolog, *No affiliation*, Obtained from: http://www.doc.gold.ac.uk/ mas02gw/prolog_tutorial/prologpages/