# BTP Report

[1]Shalini Maiti

*J-212, HOR(Women), DA-IICT Campus*

[1]201001073@daiict.ac.in

*Supervisor*

*Professor Nitin Raje*

***Abstract* – In this project, I have aimed at providing a comprehensive analysis of the different paradigms of artificial intelligence in natural language processing and conversations and how they contributed to the development of my algorithm of a conversational "bot" with which we may converse using written text.**

***Keywords* – Artificial intelligence, linguistic bots, paradigms of AI, natural language processing, conversational bots**

## I. INTRODUCTION

My objective, initially, was to research about whether or not creating a program or a "bot" that could hold a meaningful and slightly less boring conversation was possible. I read through and experimented with bots like ALICE and SIRI. Both of which are fairly efficient "bots" that are available on the open source and thus made for convenient studies. While these lacked originality, knowledge, compassion or character, it had an impeccable understanding of the English language and grammar.

This made me realise I needed to figure out what a person expects out of a conversation with someone, if a human's thought process can be formalised into an algorithm in a specific context , whether or not formalising the same is important, the essence behind thoughts, emotions, reactions and contexts.

Artificial intelligence has existed since before we could track it down in the timeline, making tremendous strides since early 1900s. So, the next step was to follow the development of AI from then until present times, observe how the different problems of AI have been handled and why they have been replaced. It was important to know why certain programs were considered to be milestones even though they are obsolete now and why they are obsolete. This is the premise of my project- understanding conversations and conversational AI, learning from the existing formalisations and programs and trying to better them according to my expectations from the same.

The paper would pan out in the following manner:

First, I will write about a human thought processes and expectations out of a conversation.

Next, I will write about the obsolete as well as existing AI programs, their contributions to my project and their flaws.

Next, I will write about my algorithm and compare it with the other existing open source algorithms, i.e, ALICE and SIRI.

## II. INTELLIGENCE AND CONVERSATIONS

### *A) What is intelligence?*

"A very general mental capability that, among other things, involves the ability to reason, plan, solve problems, think abstractly, comprehend complex ideas, learn quickly and learn from experience. It is not merely book learning, a narrow academic skill, or test-taking smarts. Rather, it reflects a broader and deeper capability for comprehending our surroundings—"catching on," "making sense" of things, or "figuring out" what to do."

### *B) What is Artificial Intelligence?*

The branch of computer science that deals with studying and creating rational agents or intelligent machines by enabling them to emulate certain traits of an intelligent being- problem solving, reasoning, perception, communication, learning, planning and movement.

### *C) Our aim and expectations?*

We shall deal with understanding, reasoning and communication.

There is a certain expectation that every person has from a conversation on the basis of the entity that is at the other end. These involve the following:

*I)* Real-time response: You can't wait for a very long time to extract a response.

*II)* Relevance: The response needs to be relevant to the context. The other person needs to understand and respond accordingly.

*III)* Spontaneity/creativity/originality or the lack of it: Depending on the relationship with the person, you either want to be surprised or not.

*IV)* Asking questions: A good conversationalist asks relevant questions.

*V)* Knowledge/Learning/Memory: Having a conversation with an amnesiac would mean having the same conversation over and over again.

If human thought process could be formalized, it would take away a very essential element of a conversation, i.e, randomness, unpredictability. The

deeper one gets into a conversation, more different a response is expected to vary from person to person. Besides, people change continually. They change with every piece of knowledge they gain. As they change, their thoughts change and thus their response to the same thread of conversation should change. Thus, it is no use trying to formalize thought processes for the development of a successful conversational "bot", we need to leave some space for randomness. Knowledge is the most basic requirement of a conversation. If I don't have knowledge of a common language, I cannot hold a conversation for very long. What we can do is mimic being in a conversation, keep learning and increasing conversational capabilities.

## III. Important Artificial Intelligence Programs

The following programs have all been studied in either of the two languages, common lisp or prolog. In this section, I will talk about the programs briefly, their methodology, what their flaws were and how they contributed to my project.

### A. *General Problem Solver(GPS)*

*I)* *Description:* it is a heuristic means-ends analysis program. It classifies things in terms of the functions they serve, functions that are required and the means to perform them. If there are more than one means, it considers the one that is most cost effective and accurate.

*II)* *Specification:* Define the current state, goal state, list of allowable operators and preconditions. If the goal condition is already in the current state, return true. Else apply appropriate operator. Update current state. If preconditions aren't fulfilled, return false.

*III)* *Analysis:* AI programming is majorly exploratory in nature. General Problem Solver did not turn out to be as general as promised. A change of domain implied adding operators. GPS V2.0 handled a lot of the problems that V1.0 faced liked the "running around the block" problem and the "recursive subgoal" problem. NP-hard problems lingered, amongst others.

*IV)* *Contribution to the project:* important method of problem solving brought into the picture, i.e, means-ends analysis. A conversation often involves a person asking a doubt and the other person trying to solve it We usually think of solutions in this manner.

### B. *ELIZA(Dialogue with a machine)*

*I)* *Description:* Held conversation with a user. Joseph Weizenbaum designed this program to be instructed in a variety of scripts, i.e, English, German and Welsh. It emulated a Rogerian psychoanalyst; it was non-directive. Patient needed to reveal oneself. This program did not truly understand language but tapped the understanding by carefully recognising, transforming, and echoing pieces of the input.

*II)* *Specification:* First, the program reads an input. Next, it finds a pattern matching the input. Then, it transforms the input appropriately into a response. Lastly, it prints out the response. The main trick here is to find the best match in the least amount of time. Firstly, separate the variables from the literals and return a list or a table for the variables. Next, use a rule-based translator. This basically acts like a pattern-matcher. If the translator sees a specific pattern, it generates an appropriate response to it from the list of responses using priority.

*III)* *Analysis:* This is a primitive way o conversing with someone. ELIZA does not truly have an understanding of the language or even the conversation. It just tricks a person into thinking it is by trying to ask relevant questions.

*IV)* *Contribution to the project:* ELIZA brought in the techniques of pattern matching and rule-based translation. If a person can find the correct rules, a language can be mapped and pattern matching could be much more accurate.

### C. *STUDENT*

*I)* *Description:* Daniel Bobrow wrote a program based on pattern-matching that could solve high school algebra word problems.

*II)* *Specification:* The program breaks input into phrses in the form of LHS=RHS, or to represent an equation. Breaks these down until one reaches numbers and variables. Translates each phrase into a math expression using a rule-based translator. Finds the equation with exactly one occurrence of an unknown, evaluates the RHS after shifting unknown on the LHS, substitutes the values in the rest of the equations, solves for each variable and prints all the values.

*III)* *Analysis:* The program can't process two different names referring to the same quantity properly. Linear equations with more than one variable can't be solved. Dividing by zero is not handled.

*IV)* *Contribution to the project:* This is also based on pattern matching and rule-based translation. However, it is more sophisticated than ELIZA. The solutions to the queries are more accurate. This is also because the way in which a high school algebra problem is written is much less ambiguous than a real-world problem or conversation. Thus, the less ambiguous input is, the better it would be understood, matched and responded with. Also, if the rules are clearly defined using grammar and semantic rules as far

as possible, responses would be much more accurate.

D. *E-MYCIN(Expert Systems)*

I) *Description:* "One that solves problems by applying knowledge that has been garnered from experts in a field." Since experts in every field need not necessarily be programmers, they can't always express language in terms to be easily translated into programs. So, these are flexible enough to handle expert knowledge and still being capable of being manipulated by programs to come up with solutions.

E-MYCIN was used to identify bacteria causing severe infections, such as bacteremia and meningitis, and to recommend antibiotics with the dosage adjusted for patient's body weight — the name derived from the antibiotics themselves, as many antibiotics have the suffix "-mycin". The Mycin system was also used for the diagnosis of blood clotting diseases.

II) *Specification:* EMYCIN=Prolog+ uncertainty factors+ caching + asking questions + explanations + contexts – variables.

This replaces Boolean with certainty fctors to account for a more flexible range, as exists in a real-world scenario. It caches all the facts it derives in an database, deals with instances of objects and their attributes. Three other sources of information also have to be maintained-rule base(defined by experts), structures for parameter definition and control flow(managed by list of contexts). Context is defined as "a situation within which the program responds." Technically, it is like a data type.

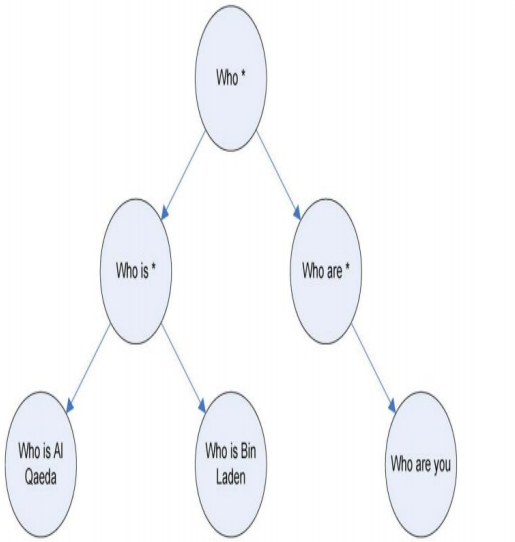
III) *Analysis:* Emycin was a very successful program. It did better at diagnosing than the Stanford medical school faculty. However, the inherent issue that patients and the society in general had with having a computer diagnose them was, in case of an error, who would take responsibility. It was this accurate because of its use of perfectly defined rules, contexts in which the program gained perfect knowledge from experts in the field.

IV) *Contribution to the project:* The most important contribution that expert systems had towards the project is that context-specific knowledge is extremely important to hold a conversation. Also, when the program does not know, it learns by asking questions, thus gaining the bit of knowledge that it did not have, initially. This may not be a generalised problem solver or a great conversationalist in a different context, but it is perfectly well-equipped to respond accurately to the conversation that its target audience would involve it in.

E. *CASE BASED REASONING( ALICE):*

I) *Description:* A.L.I.C.E. is a Virtual Character based on AI system. It is not based on complicated AI mechanisms like neural network, knowledge representation, search, fuzzy logic, genetic algorithms or parsing. The theory that is behind its AI is called "Case-Based Reasoning" or CBR.

II) *Specification:* The theory that describes the A.L.I.C.E. algorithm is called "Case-Based Reasoning" or CBR. The CBR "cases" are the categories in AIML. The algorithm finds best-matching pattern for each input. Thus A.L.I.C.E. learns and thinks.




III) *Analysis:* ALICE uses XML-based Artificial Intelligence markup Language (AIML) files to hold its internal collection of knowledge. This open-source knowledge base makes ALICE robust and able to quickly extend into new knowledge domains. ALICE seeks to mimic conversation rather than understand it.

IV) *Contribution to the project:* This is a fairly successful bot. Using a very high number of cases, a huge database and an excellent search algorithm, we are able to provide the program a lot of knowledge and information. Since this is not context-specific, it has to sift faster through its cases and find the best match. It asks questions and learns.

## IV. THE FRIEND'S ALGO

A. *How it works:*

A script with more than one characters is fed into the database and each related thread is given a probability

according to the number of words in the query. Also, in the same conversation, each response is linked to another by the same formula. When a user types in a query, they are matched in the database and the response with the highest probability is returned or printed. The database is a simple sql database. I have used Action script as the programming language and Flash as IDE.

In this database, I have fed in 2 seasons of a long lasting T.V series called F.R.I.EN.D.S with surrounds the lives of six major characters, all of whom have different ways of thinking and responses.

What this algorithm ensures, is a context and yet randomness in the responses. Since, a person has built these characters and given them dialogues, it allows for a user to feel like he or she is talking to a regular person.

It is inspired from ELIZA, Expert systems and ALICE; wherein it uses word matching, instead of pattern matching, and works on being fed as much information about a character as possible by the one that has created it. We chose this series because it has 10 seasons, 6 major individualistic characters and a lot of conversations. Only 2 seasons have been added into the database because the rest were unavailable. But, once they are fed in as well, it would ensure more uniform probability distribution.

*B. The algorithm:*

The algorithm shall be explained using examples:

Suppose, there is human conversation between Chandler and Joey.

Chandler : I want burger
Joey: Do you like it with cheese ?

Chandler : I want money
Joey: Me too, everyone wants

Chandler : MacD cheese burger would be good ?
Joey: I think so

Now, here if we take the first line as the query question, and second lines as answer to those queries, we could make database as shown below.

For,
**Chandler : I want a burger**
**Joey : Do you like it with cheese ?**

In the question, there are 3 words, 'I', 'want', 'burger'. So if we link it's answer and provide weightage to each, our database will be like,

I -> [(Do you like it with cheese ?, 1/3)]
Want -> [(Do you like it with cheese ?, 1/3)]
Burger -> [(Do you like it with cheese ?, 1/3)]

**Chandler : I want money**
**Joey : Me too, everyone wants**

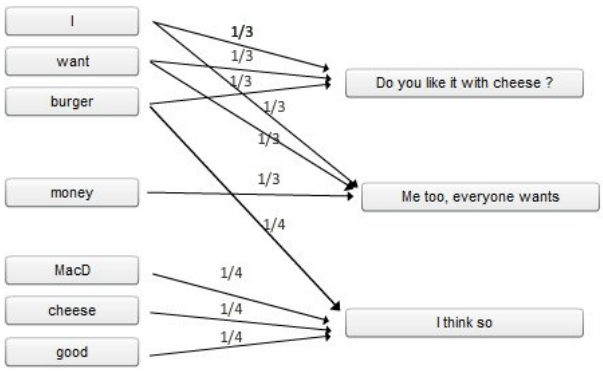
In our database, we already added 'I' and 'Want'. We have to add 'Money' only as new word and provide weightage to according words.

I -> [(Do you like it with cheese ?, 1/3), (Me too, everyone wants , 1/3)]
Want -> [(Do you like it with cheese ?, 1/3), (Me too, everyone wants , 1/3)]
Burger -> [(Do you like it with cheese ?, 1/3)]
Money -> [(Me too, everyone wants , 1/3)]

**Chandler : MacD cheese burger would be good ?**
**Joey : I think so**

MacD, cheese, burger, good (New Words)

I -> [(Do you like it with cheese ?, 1/3), (Me too, everyone wants , 1/3)]
Want -> [(Do you like it with cheese ?, 1/3), (Me too, everyone wants , 1/3)]
Burger -> [(Do you like it with cheese ?, 1/3), I think so , 1/4)]
Money -> [(Me too, everyone wants, 1/3)]
MacD -> [(I think so, 1/4)]
Cheese -> [(I think so, 1/4)]
Good -> [(I think so, 1/4)]




**Original Conversation :**

Chandler : I want burger
Joey: Do you like it with cheese ?

Chandler : I want money
Joey: Me too, everyone wants

Chandler : MacD cheese burger would be good ?
Joey: I think so

So, whenever the user enters query, it will look for related words from database and whichever sentence has highest weightage, the "bot" will reply with that answer.

Suppose, if we have database as shown below,

I -> [(Do you like it with cheese ?, 1/3), (Me too, everyone wants , 1/3)]
Want -> [(Do you like it with cheese ?, 1/3), (Me too, everyone wants , 1/3)]
Burger -> [(Do you like it with cheese ?, 1/3), I think so , 1/4)]
Money -> [(Me too, everyone wants, 1/3)]
MacD -> [(I think so, 1/4)]
Cheese -> [(I think so, 1/4)]
Good -> [(I think so, 1/4)]

And if user enters query like,

"I want a good cheese burger."

Algo will look for related words (I, want, good, cheese, burger) and will fetch related sentences and sums up related weightages,

I -> [(Do you like it with cheese ?, 1/3), (Me too, everyone wants , 1/3)]
Want -> [(Do you like it with cheese ?, 1/3), (Me too, everyone wants , 1/3)]
Good -> [(I think so, 1/4)]
Cheese -> [(I think so, 1/4)]
Burger -> [(Do you like it with cheese ?, 1/3), I think so , 1/4)]

So, final weightages for given sentences will be,

1. (Do you like it with cheese ?) -> 1/3 + 1/3 + 1/3 = 1
2. (Me too, everyone wants) -> 1/3 + 1/3 = 2/3
3. (I think so) -> 1/4 + 1/4 + 1/4 = 3/4

So, here sentence 1 has highest weightage. So bot will reply with first sentence.

Now, for more accuracy if we add one more factor, 'total weightage' for every word. For example, 'I' word has two sentences linked and each has weightage of '1/3' so, total weightage of 'I' will be 1/3 + 1/3 = 2/3. If we calculate total weightage for every word, we will have new database as,

I (2/3) -> [(Do you like it with cheese ?, 1/3), (Me too, everyone wants , 1/3)]
Want (2/3) -> [(Do you like it with cheese ?, 1/3), (Me too, everyone wants , 1/3)]
Burger (7/12) -> [(Do you like it with cheese ?, 1/3), I think so , 1/4)]
Money (1/3) -> [(Me too, everyone wants, 1/3)]
MacD (1/4) -> [(I think so, 1/4)]
Cheese (1/4) -> [(I think so, 1/4)]
Good (1/4) -> [(I think so, 1/4)]

So, now if user enters the same query, for calculating weights of sentences we will consider individual weights and related words' total weights and multiply them for the final weight. For example,

I (2/3) -> [(Do you like it with cheese ?, 1/3), (Me too, everyone wants , 1/3)]
Want (2/3) -> [(Do you like it with cheese ?, 1/3), (Me too, everyone wants , 1/3)]
Good (1/4) -> [(I think so, 1/4)]
Cheese (1/4) -> [(I think so, 1/4)]
Burger (7/12) -> [(Do you like it with cheese ?, 1/3), I think so , 1/4)]

1. (Do you like it with cheese ?) -> (2/3)(1/3) + (2/3)**(**1/3) + (7/12)**(**1/3) = 23/36 = 0.6389
2. (Me too, everyone wants) -> (2/3)**(**1/3) + (2/3)**(**1/3) = 4/9 = 0.4444
3. (I think so) -> (1/4)**(**1/4) + (1/4)**(**1/4) + (7/12)**(**1/4) = 13/48 = 0.2708

So, the bot will reply with sentence 1 as sentence 1 has the highest weight.

*C.* Limitations

I) This bot cannot handle queries including words that are not already in the database.
II) This is not a learning bot. Hence, only existing responses would be provided s output in the conversations.

*D.* Future Scope

I) This bot could take input from the user, check for relevance, and feed it into the database. For that, it would be essential for the bot to have knowledge of natural language syntax and semantics.
II) Use more scripts or expand the same to populate the database for more queries.
III) Make character-wise, context-based bots by using two more parameters.
IV) Use dictionary to handle queries with words that are synonymous but don't exist in the scripts already provided.
V) Use the bots to emulate a person in applications like character-centric games or expert systems where a database can be populated with perfect information about the person that the bot is trying to emulate.

## V. CONCLUSIONS

A conversation always has a context. Thus, a conversation can only ensue for as long as both sides have relevant knowledge

or as long as either side has the knowledge and the other one is ready to learn.

A conversational “bot” may respond accurately even without a thought process as long as it has enough cases and a good search algorithm. Thus, it is important to mimic human conversations rather than trying to formalize human thought process.

This bot can be applied in applications where a character has to behave in a certain manner according to a context, where the person has a set number of things to say. This can also be used like the expert systems have. Expert systems use certainty factors and this algorithm uses weighted probabilities. Basically, wherever a person can provide a certain character with near-perfect information about itself, this bot can made use of. Since, it can emulate an intelligent being, this bot can hold conversations in place of a person. Like ALICE an SIRI, this would also be made open source for people to be able to use it in such applications and add to it.


## ACKNOWLEDGMENT

I acknowledge the institute, DA-IICT and my B.Tech project mentor, Professor Nitin Raje for guiding me through every aspect of this undertaking.


## REFERENCES


[1] Mainstream Science on Intelligence" (1994),

[2] Towards Artificial Sapience(Rene V. Mayorga, Leonid I. Perlovsky).

[3] Conversational Virtual Character for the Web(Karlo Smid, Igor S. Pandzic)

[4] Paradigms of Aritficial Intelligence Programming: Case Studies in Common Lisp(Peter Norvig)