

# **Intello-An Intelligent Chatbot**

*Project Submitted in fulfilment of the requirements of*

**BACHELOR OF TECHNOLOGY**



**Shaheed Sukhdev College of Business Studies**

**University of Delhi, Delhi**

April 2017

## **Project Report**

### **Intello – An Intelligent Chatbot**

Work Carried Out

*By*

*Arjun Malhotra, Gunjan Gupta, Khushboo Chitre, Nitesh Kumar Gupta and Ojasvi Aggarwal*

*Under the Supervision of*

**Dr. Anamika R. Gupta**

Assistant Professor

Department of Computer Science

Shaheed Sukhdev College of  
Business Studies

University of Delhi

*Submitted in fulfilment of the requirements of*

**BACHELOR OF TECHNOLOGY**

**To**

*Shaheed Sukhdev College of Business Studies*

*University of Delhi*

*April 2017*



## CERTIFICATE

This is to certify that the project work entitled “**Intello - An Intelligent Chatbot**” has been carried out by **Arjun Malhotra, Gunjan Gupta, Khushboo Chitre, Nitesh Kumar Gupta and Ojasvi Aggarwal** under the supervision of **Dr. Anamika R. Gupta**, Assistant Professor in Department of Computer Science, Shaheed Sukhdev College of Business Studies, University of Delhi. This work has been carried out for the fulfilment of the requirements of the B.Tech. (Bachelor of Technology) degree in Department of Computer Science, University of Delhi. This project has not been submitted anywhere for the purpose of any other degree or diploma.

**Signature:**

**Signature:**

**Dr. Anamika R. Gupta**  
Assistant Professor  
Department of Computer Science  
Shaheed Sukhdev College of  
Business Studies  
University of Delhi

**Dr. Ajay Jaiswal**  
Head of Department  
Department of Computer Science  
Shaheed Sukhdev College of  
Business Studies  
University of Delhi

## **ACKNOWLEDGEMENT**

We have made efforts in this project to complete it in the best way possible. However, it would not have been possible without the kind support and help of many people. We would like to extend our sincere gratitude to all of them.

We are highly indebted to Dr. Anamika R. Gupta for her guidance and constant supervision, as well as for providing us with the necessary backdrop and content in various sections of the project.

We are also grateful to Dr. Ajay Jaiswal, Head, Department of Computer Science, for providing us with the opportunity to work on this project.

**Arjun Malhotra, Gunjan Gupta, Khushboo Chitre, Nitesh Kumar Gupta, Ojasvi Aggarwal**

B.Tech. (Computer Science)  
Shaheed Sukhdev College of Business Studies  
University of Delhi  
Delhi-110095

# TABLE OF CONTENTS

<b>Table of Contents .....</b>	<b>5</b>
<b>1. Abstract .....</b>	<b>6</b>
<b>2. Introduction.....</b>	<b>7</b>
<b>3. Related Work .....</b>	<b>8</b>
3.1 Api.ai .....	8
3.2 Wit.ai .....	9
3.3 Motion.ai.....	9
<b>4. Background .....</b>	<b>11</b>
4.1 ChatBot .....	11
4.2 Type of models used for ChatBot.....	11
4.3 Type of Conversations .....	11
4.4 Type of Platforms for chatbot .....	12
4.5 ALICE- Natural Language Processing Chatterbot.....	12
4.6 Architecture of ChatBot.....	13
4.7 Basics of AIML .....	14
<b>5. Specifications .....</b>	<b>16</b>
5.1 System features .....	16
5.2 Non-Functional Requirements .....	17
5.2.1 Performance Requirements .....	17
5.2.2 Safety Requirements .....	18
5.2.3 Security Requirements .....	18
5.2.4 Software Quality Attributes.....	18
5.2.5 Business Rules .....	18
<b>6. Design.....</b>	<b>19</b>
<b>7. Implementation .....</b>	<b>25</b>
<b>8. Testing .....</b>	<b>28</b>
<b>9. Results and analysis .....</b>	<b>31</b>
<b>10. Conclusion .....</b>	<b>32</b>
<b>11. References.....</b>	<b>33</b>

## **ABSTRACT**

A user often needs to go through each and every question of the FAQ section of a website simply to find the answer to his solitary question. It would be much simpler if there existed a single interface through which he could simply type in his question and the answer is directly provided. That is precisely what this project aims to provide. It includes the development of a chatbot that is designed to answer all queries related to the admission procedure of an educational institute. The chatbot was designed from scratch using Python and AIML. Python was used to implement the heart (the kernel). AIML was used to implement the brain (the knowledge base).

The bot was tested rigorously, not just by the development team, but also by a good sample of general users. As predicted, it was able to answer accurately almost every question about the admission procedure of Shaheed Sukhdev College of Business Studies. Moreover, the convenient admin interface makes it easier to update the knowledge base regularly. It is thus an obvious conclusion that chatbots, which are gaining traction in the e-commerce and online customer service industry, can also be useful for educational institutions.

## INTRODUCTION

A chatbot (also known as a talkbot, chatterbot, bot, chatterbox, Artificial Conversational Entity) is a computer program which conducts a conversation via auditory or textual methods.<sup>[3]</sup> Such programs are often designed to convincingly simulate how a human would behave as a conversational partner, thereby passing the Turing test.<sup>[3]</sup>

AI based chatbots are in great demand today. Since the advent of messaging apps, natural language based query processing is fast replacing static query based applications, window based applications, and customer care agents. More and more companies in the technological sector are looking to incorporate AI based chatbots into their websites and apps.

The benefit of such a system is that a machine neither gets tired, nor forgets anything. Even a smart human has the tendency to forget the nuances of certain procedures. However, everything stored in the knowledge base of a chatbot stays there forever. So, it can act as a worthy replacement for the FAQ section of a website, since an FAQ has a limited number of questions, and the user may not always find an answer to the exact question he's looking for.

The easy to maintain and update nature of a chatbot allows it better flexibility, and makes it more convenient for a user looking for answers to specific questions. It is also better than a window based android application, since such an application requires the user to find the appropriate links and drop down menus.

The aim of this project is to create an intelligent chat agent (Intello) that can potentially answer all common queries regarding the admission procedure of an educational institute. Shaheed Sukhdev College of Business Studies has been taken as an example. Intello will act as a solution provider for all queries related to the admission process of Shaheed Sukhdev college of Business Studies. The user can simply type in a query through the application, and Intello will consult its knowledge base to answer it intelligently.

To fulfill the vision we had for Intello, we chose to design it from scratch, using Python, AIML, and PHP. The main engine was designed using Python, since it is a modern programming language designed to simplify Artificial Intelligence based programs. Python was also used to implement the Text-to-Speech feature, since it provides many easy to use libraries (pyttsx) which are close to hardware. The Knowledge Base was designed using AIML, since it is extremely easy to use (resembles XML). AIML provides the easiest way to store and maintain a Knowledge Base for a chatbot. Existing chatbots use AIML too. PHP was used to design the admin interface for maintenance of the bot, since it is easy to use with HTML forms (which provide a convenient way for a layman admin to submit queries and responses).

Intello is currently in its beta stage. Many more functionalities, such as session chat history and student login (to fetch attendance) can be incorporated, to make the application even more useful. Moreover, the knowledge base can be updated with questions based on more and more domains, other than the ones we have already covered. At this stage, Intello is a conversational bot that can replace the FAQ on the college website. With further development, it could potentially make the college website much more convenient and fun to use.

## RELATED WORK

Many chatbot development engines are available on the internet today. Wit.ai, Motion.ai and Api.ai are just three of the various tools that are currently available in the chatbot development market. Each of these is built for specific test cases, and based on the type of chatbot required, one or the other may be more appropriate.

### Api.ai

Intents and Contexts are the key concepts to model the behaviour of a chatbot with Api.ai.<sup>[34]</sup> Intents create links between what a user says and what action should be taken by the bot.<sup>[34]</sup> Contexts are string values, useful for differentiating requests which might have different meanings depending on previous requests.<sup>[34]</sup>

#### Features

- Api.ai has a paid enterprise option which allows for this to be run on a private cloud internally and more from their services team. Free plan allows 6000 requests per month.<sup>[5]</sup>
- It provides speech to text and text to speech capabilities along with machine learning.<sup>[5]</sup>
- It has support for intents, actions, and entities along with domains.<sup>[5]</sup>

#### Pros

- Api.ai proposes a powerful way of modelling large and complex flows using Intents and Contexts.<sup>[34]</sup>
- Slot-filling is an integrated feature.<sup>[34]</sup> Consequently, a good part of the logic can be solved by the chatbot, which decreases the server side coding.<sup>[34]</sup>
- Domains are available, that is, specifications that can deal with several common use cases and applications (e.g. small talk, wisdom, flight schedules, reminders, etc).<sup>[34]</sup>
- A section “Training” (in beta) is proposed to train the chatbot with examples.<sup>[34]</sup>
- One-click integration with several platforms: Facebook Messenger, Slack, Twitter, Telegram, etc.<sup>[34]</sup>

#### Cons

- It is impossible to block the matching of intent if a context is present.<sup>[34]</sup> For example, if the word ‘pizza’ is an intent and ‘ordering’ is a context, any sentence that contains the word ‘pizza’ and ‘order’ will be matched to the context ‘ordering’ with the intent ‘pizza’, even if the sentence says, “I had ordered my brother to get me a pizza last night, but he didn’t do it.” Consequently, the reply might be, “Thank you for ordering a pizza.”<sup>[34]</sup>
- You cannot model an intent that can be matched only if a certain context is not present.<sup>[34]</sup> For example, if a person is allowed to have either a pizza, or a burger, but not both, then you cannot model the chatbot to distinguish between the two cases.<sup>[34]</sup> If a user first asks for a pizza, and then later asks for a burger, there is no way that the chatbot can say, “I’m sorry. You cannot have a burger.”<sup>[34]</sup>
- The training section is still in beta.<sup>[34]</sup>



## **Wit.ai**

Stories are the key concept to model the behaviour of a chatbot with Wit.ai.<sup>[34]</sup> Each story represents an example of a possible conversation.<sup>[34]</sup> It should be noted that “intent” is no longer a concept but a user entity, non-mandatory.<sup>[34]</sup> This was a change of greater impact in Wit.ai, motivated by the fact that a complex chatbot needs a lot of intents that can, in some way, be grouped into stories.<sup>[34]</sup>

### **Pros**

- The concept of stories is powerful.<sup>[34]</sup>
- Wit.ai allows controlling the conversation flow using branches and also conditions on actions (e.g., show this message only if some specific variables are defined).<sup>[34]</sup>
- Assigning roles to entities helps server side processing.<sup>[34]</sup>
- A section “Understanding” is proposed to train the chatbot with examples.<sup>[34]</sup>
- An “Inbox” exists, where the requests that could not be processed by the chatbot are listed, so the developers can teach the bot.<sup>[34]</sup>

### **Cons**

- Stories are in beta.<sup>[34]</sup>
- Even if stories are a powerful concept, there are cases where it is difficult to control the flow of the conversation and the bot tends to misunderstand the user requests.<sup>[34]</sup>
- When a question like “How many people signed up in October?” is asked, it defaults to assuming that October refers to the ‘coming October’. So, such questions cannot be answered accurately.<sup>[34]</sup>

## **Motion.ai**

It provides a platform that gives anyone the ability to easily build and deploy a chatbot on a variety of different mediums.<sup>[35]</sup> Conversations and reports via a Motion.ai chatbot allow developers to see how users are interacting with the bot.<sup>[35]</sup>

### **Pros**

- Provides ongoing analytical and reporting tools.<sup>[35]</sup> Motion.ai also handles things like analytics and reports on how the bot is being used after deployment.<sup>[35]</sup>
- It includes what it calls “modules,” which package up the logic required for building particular bot features.<sup>[35]</sup> It is done by offering “Yes/No,” “Multiple Choice,” and a series of other modules that send and receive data.<sup>[35]</sup>
- It is good for scripted channels and conversations as it works on the basis of providing options to choose from to the users.<sup>[35]</sup>
- Motion.ai helps users create connections between modules with the help of flowcharts.<sup>[35]</sup>
- In Motion.ai, module connections are developed using if statements and the conditions for the if statements depend upon two factors-

- Extracted data- Some modules will parse and extract data from users' replies.<sup>[35]</sup> This extracted data shows up on Conversations and Reports within any module.<sup>[35]</sup>
- Custom variables- Custom variables are key-value pairs like (custom Var KEY=VALUE), where KEY is the name of the variable and VALUE is its contents.<sup>[35]</sup>
- It automatically deals with spelling errors.<sup>[35]</sup>
- Motion.ai provides various response formatting options like-
  - Randomized responses so that users have a feeling of talking to a human.<sup>[35]</sup>
  - Embedded media (images/audio/video) within responses using media URL.<sup>[35]</sup>
  - Referencing prior responses so that in the case of randomized responses, some responses get priority over others in some situations.<sup>[35]</sup>
  - Breaking responses into multiple chat bubbles.<sup>[35]</sup> This also gives a user the feeling of talking to a human, as a human being doesn't quote complete response in one statement.<sup>[35]</sup>

## Cons

- Motion.ai only operates on either a multiple choice or bot statement basis.<sup>[35]</sup>
- Users can't specify questions that might get asked to the bot and replied with proper responses.<sup>[35]</sup> This happens because Motion.ai works by providing multiple choices to the users to choose from. So, users can't quote their own queries.<sup>[35]</sup>
- It wouldn't be very good for any platform where a freer format of conversation is anticipated.<sup>[35]</sup>

## BACKGROUND

### Chatbot

Short for chatterbot, a computer program that simulates human conversation, or chat, through artificial intelligence.<sup>[3]</sup> Typically, a chat bot will communicate with a real person, but applications are being developed in which two chat bots can communicate with each other.<sup>[3]</sup>

Chatterbots are typically used in dialog systems for various practical purposes including customer service or information acquisition.<sup>[3]</sup> Some chatterbots use sophisticated natural language processing systems, but many simpler systems scan for keywords within the input, then pull a reply with the most matching keywords, or the most similar wording pattern, from a database.<sup>[3]</sup>

The term "Chatterbot" was originally coined by Michael Mauldin (creator of the first Verbot, Julia) in 1994 to describe these conversational programs.<sup>[3]</sup> The classic historic early chatbots are ELIZA (1966) and PARRY (1972).<sup>[3]</sup> More recent notable programs include A.L.I.C.E., Jabberwacky, and D.U.D.E. (Agence Nationale de la Recherche and CNRS 2006).<sup>[3]</sup>

### Type of models used for Chatbot

- **Retrieval-based models** (easier) use a repository of predefined responses and some kind of heuristic to pick an appropriate response based on the input and context.<sup>[1]</sup> The heuristic could be as simple as a rule-based expression match, or as complex as an ensemble of Machine Learning classifiers.<sup>[1]</sup> These systems don't generate any new text. They just pick a response from a fixed set.<sup>[1]</sup>
- **Generative models** (harder) don't rely on pre-defined responses.<sup>[1]</sup> They generate new responses from scratch.<sup>[1]</sup> Generative models are typically based on Machine Translation techniques, but instead of translating from one language to another, they "translate" from an input (query) to an output (response).<sup>[1]</sup>

### Type of conversations

- **Short-Text Conversations** (easier) where the goal is to create a single response to a single input.<sup>[1]</sup> For example, you may receive a specific question from a user and reply with an appropriate answer.<sup>[1]</sup>
- **Long conversations** (harder) where you go through multiple turns and need to keep track of what has been said.<sup>[1]</sup> Customer support conversations are typically long conversational threads with multiple questions.<sup>[1]</sup>
- In an **open domain** (harder) setting the user can take the conversation anywhere.<sup>[1]</sup> There isn't necessarily a well-defined goal or intention.<sup>[1]</sup> Conversations on social media sites like Twitter and Reddit are typically open domain—they can go into all kinds of directions.<sup>[1]</sup> The infinite number of topics and the fact that a certain amount of world knowledge is required to create reasonable responses makes this a hard problem.<sup>[1]</sup>

- In a **closed domain (easier)** setting the space of possible inputs and outputs is somewhat limited because the system is trying to achieve a very specific goal.<sup>[1]</sup> Technical Customer Support or Shopping Assistants are examples of closed domain problems.<sup>[1]</sup> These systems don't need to be able to talk about politics, they just need to fulfil their specific task as efficiently as possible.<sup>[1]</sup> Sure, users can still take the conversation anywhere they want, but the system isn't required to handle all these cases—and the users don't expect it to.<sup>[1]</sup>

## Type of platforms for Chatbots

- **Goal-oriented or transactional:** It is the most frequent kind of chatbot for business.<sup>[4]</sup> It helps user achieve tasks such as buying a ticket, ordering food or getting specific information.<sup>[4]</sup>
- **Conversational platforms:** The main goal here is to allow the user to have a conversation with the bot, without considering a task-oriented scenario.<sup>[4]</sup> These platforms typically use specification languages such as AIML (Artificial Intelligence Markup Language) to model the interactions with the user.<sup>[4]</sup>  
When the user says “my dog’s name is Max”, the chatbot recognizes that pattern and extracts the dog’s name.<sup>[4]</sup> It must be noted that this extraction by text match is very simple if we compare it with the power of NLP information extraction.<sup>[4]</sup> The chatbot is going to respond with “That is interesting that you have a dog named Max”.<sup>[4]</sup> Later, if the user asks for his dog’s name, the chatbot would be able to respond “Your dog’s name is Max”.<sup>[4]</sup>

### Pros

- AIML is a standard.<sup>[4]</sup>
- It is very flexible to create conversations.<sup>[4]</sup>

### Cons

- It could be difficult to scale if patterns are manually built.<sup>[4]</sup>
- The information extraction capabilities are limited.<sup>[4]</sup>
- They are not really appropriate for task oriented bots.<sup>[4]</sup>

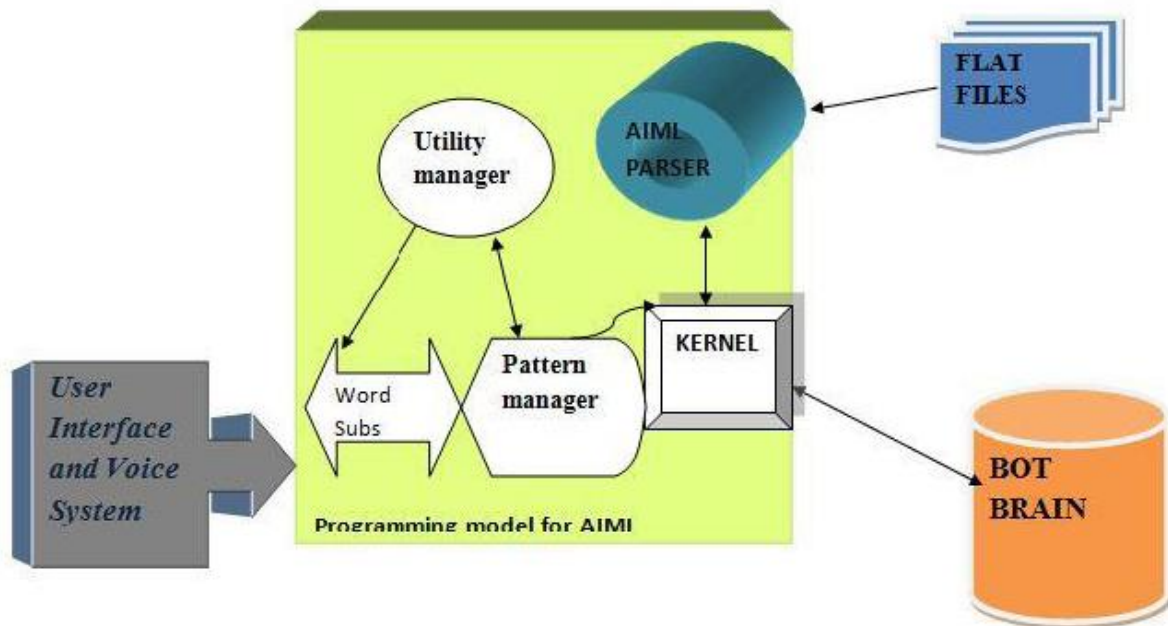
## ALICE - Natural Language Processing Chatterbot

A.L.I.C.E. (Artificial Linguistic Internet Computer Entity), also referred to as Alicebot, or simply Alice, is a natural language processing chatterbot—a program that engages in a conversation with a human by applying some heuristical pattern matching rules to the human's input, and in its online form it also relies on a hidden third person.<sup>[32]</sup> It was inspired by Joseph Weizenbaum's classical ELIZA program.<sup>[32]</sup> It is one of the strongest programs of its type and has won the Loebner Prize, awarded to accomplished humanoid, talking robots, three times (in 2000, 2001, and 2004).<sup>[32]</sup> The software used to create A.L.I.C.E. is available as free ("open source") Alicebot and AIML software.<sup>[33]</sup>

## Architecture of a Chatbot

The bot engine is the main architecture which consists of the following components<sup>[6]</sup>:

- a) Kernel<sup>[6]</sup>
- b) AIML Parser<sup>[6]</sup>
- c) Utility Manager<sup>[6]</sup>
- d) Word Substitution<sup>[6]</sup>
- e) Pattern Manager<sup>[6]</sup>



### Kernel

The Kernel class has most of the implementation and is responsible for all processing.<sup>[6]</sup> It requires all other classes for the processing to happen.<sup>[6]</sup> It requires AIML parser, word substitution, utility manager and pattern manager for matching.<sup>[6]</sup> It includes modules like sessions, bot predicates, word substitute, and element processor.<sup>[6]</sup> It also includes the brain file. If a brain file is provided, the kernel tries to load it.<sup>[6]</sup> If it is not provided, then it attempts to load all AIML files.<sup>[6]</sup>

### AIML Parser

AIML parser is just like an xml parser.<sup>[6]</sup> It gets the name of the current unknown element, and then counts the number of errors occurred in the AIML document to test the authenticity of document.<sup>[6]</sup> If the number of errors occurred is greater than the threshold value, the AIML file is skipped.<sup>[6]</sup>

### Utility Manager

It includes the assorted utility function which is used in PyAIML package for performing various functions.<sup>[6]</sup> Its task is to split a given string into a list of sentences.<sup>[6]</sup> Therefore, it

helps in splitting the input string, thus helping the AIML parser perform all the required processing.<sup>[6]</sup>

### **Word Substitution**

It implements the word substitution class adopted from python cookbook module for simplicity and accuracy.<sup>[6]</sup> This class is used like a dictionary to add pairs of words anywhere in the dictionary.<sup>[6]</sup> It also contains another method called “substitution method” for substituting words in the string. All matching done for replacement is case sensitive and intelligent.<sup>[6]</sup>

### **Pattern Manager**

This class implements the pattern matching algorithm of AIML, which makes pattern matching more intelligent and accurate.<sup>[6]</sup> The algorithm follows like this:

- The number of templates currently stored is calculated.<sup>[6]</sup>
- The name of the chatbot for its uniqueness is set.<sup>[6]</sup> It must consist of a single word.<sup>[6]</sup>
- If any pattern is found, it is dumped for debugging purposes.<sup>[6]</sup>
- If any further pattern is found, it is stored in a file specified like a pickle file.<sup>[6]</sup>
- After all patterns are stored, they are restored for making a brain map.<sup>[6]</sup>
- The brain map is used to interact with the user.<sup>[6]</sup>

### **Basics of AIML**

AIML stands for Artificial Intelligence Modelling Language.<sup>[2]</sup> AIML is an XML based markup language meant to create artificial intelligent applications such as Chatbots.<sup>[2]</sup> AIML makes it possible to create human interfaces while keeping the implementation simple to program, easy to understand and highly maintainable.<sup>[2]</sup> AIML was developed by the Alicebot free software community and Dr. Richard S. Wallace during 1995-2000.<sup>[2]</sup> AIML is used to create or customize Alicebot which is a chat-box application based on A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) free software.<sup>[2]</sup>

Following are the important tags which are commonly used in AIML documents.<sup>[2]</sup>

<aiml> : Defines the beginning and end of a AIML document.<sup>[2]</sup>

<category> : Defines the unit of knowledge in Alicebot's knowledge base.<sup>[2]</sup>

<pattern> : Defines the pattern to match what a user may input to an Alicebot.<sup>[2]</sup>

<template> : Defines the response of an Alicebot to user's input.<sup>[2]</sup>

<star> : Used to match wild card \* character(s) in the <pattern> Tag.<sup>[2]</sup>

<srai> : Multipurpose tag, used to call/match the other categories.<sup>[2]</sup>

<random> : Used <random> to get random responses.<sup>[2]</sup>

<li> : Used to represent multiple responses.<sup>[2]</sup>

<set> : Used to set value in an AIML variable.<sup>[2]</sup>

<get> : Used to get value stored in an AIML variable.<sup>[2]</sup>

<that> : Used in AIML to respond based on the context.<sup>[2]</sup>

<topic> : Used in AIML to store a context so that later conversation can be done based on that context.<sup>[2]</sup>

<think> : Used in AIML to store a variable without notifying the user.<sup>[2]</sup>

<condition> : Similar to switch statements in programming language. It helps ALICE to respond to matching input.<sup>[2]</sup>

AIML vocabulary uses words, space and two special characters \* and \_ as wild cards.<sup>[2]</sup>

AIML interpreter gives preference to patterns having \_ over patterns having \*.<sup>[2]</sup> AIML tags are XML compliant and patterns are case-insensitive.<sup>[2]</sup>

Example <sup>[2]</sup>

```
<aimlversion="1.0.1"encoding="UTF-8"?>
<category>
<pattern> HELLO ALICE </pattern>

<template>
    Hello User!
</template>

</category>
</aiml>
```

## SPECIFICATIONS

### ➤ System Features

The users are divided into four classes - Student, Faculty, Admission seekers and Knowledge base administrators. Following are the response sequences and functional requirements of the mentioned user classes.

#### User Class – Student

**Description and Priority:** The application will be most commonly and frequently used by the user class - student. This user class is expected to use most of the product features, and hence the priority level is high.

##### Response Sequence

- User enters query regarding Admission related Information.
- Bot analyses the query using exhaustive keywords and patterns stored in knowledge base (AIML Database).
- It generates and displays a relevant answer to the query.
- If the user is not satisfied with the answer, and asks the same question in a different way, the bot will look for another, more relevant answer.
- If an unexpected query is encountered, the bot will request the user to rephrase.

##### Functional Requirements

- The system should have a knowledge base, wherein exhaustive keywords and patterns are stored.
- The system should be able to understand the context of the queries.

#### User Class - Faculty

**Description and Priority:** The user class – faculty will use the application to update Attendance and Internal Assessment marks, and to read articles on the college website. The priority level is medium.

##### Response Sequences

- User enters query regarding Admission related Information.
- Bot analyses the query using exhaustive keywords and patterns stored in knowledge base (AIML Database).
- It generates and displays a relevant answer to the query.
- If the user is not satisfied with the answer, and asks the same question in a different way, the bot will look for another, more relevant answer.
- If an unexpected query is encountered, the bot will request the user to rephrase.

##### Functional Requirements

- The system should have a knowledge base, wherein exhaustive keywords and patterns are stored.
- The system should be able to understand the context of the queries.



## **User Class – Admission Seekers**

**Description and Priority:** This class includes people seeking admission, and hence will use queries related to admission procedure the most. The priority level is high.

### **Response Sequences**

- User enters query regarding Admission related Information.
- Bot analyses the query using exhaustive keywords and patterns stored in knowledge base (AIML Database).
- It generates and displays a relevant answer to the query.
- If the user is not satisfied with the answer, and asks the same question in a different way, the bot will look for another, more relevant answer.
- If an unexpected query is encountered, the bot will request the user to rephrase.

### **Functional Requirements**

- The system should have a knowledge base, wherein exhaustive keywords and patterns are stored.
- The system should be able to understand the context of the queries.

## **User Class – Knowledge Base Administrators**

**Description and Priority:** This class will include people responsible to manage and update the knowledge base, and hence will not directly interact with the application. The priority level is high.

### **Response Sequences**

- User logs in and updates the knowledge base, as per feedback, or accordingly.
- The bot will get access to the updated knowledge base, and will be able to handle new queries.

### **Functional Requirements**

- The system should have asked the user to log in to his account.
- The admin must have access to manipulate the DB.
- The system should maintain a file of unexpected queries, so that the admin can update the knowledge base easily.

### **➤ Non-functional Requirements**

### **Performance Requirements**

- The clients' user interface should be compatible with all commonly used browsers, such as Internet explorer, Firefox, Google Chrome and Safari.
- The system shall be able to scale based on the number of users using the system.
- Long answers (15 words or more) should be returned in the range of six to ten seconds while short answers (less than 15 words) should be returned in less than five

seconds, to give the user the feeling of interacting with a human being rather than a bot.

- The system should run on a variety of operating systems that support the Python, AIML, and PHP.
- The system should run on a variety of hardware.

### **Safety Requirements**

- The system shall not store or process any information about its users.

### **Security Requirements**

- The administrative system should be protected from unauthorized access.
- The knowledge base should be protected from attacks and unauthorized access.
- The interface should be protected from attacks.
- All passwords should be stored using secure methods.

### **Software Quality Attributes**

- The system should be easy to maintain.
- The system shall maintain an easy to use interface across all functionality and for all users.
- There should be a clear separation of Python, AIML, and PHP code.
- There should be a clear separation between the interface and the business logic code.
- There should be a clear separation between the data access objects that map the Database and the business logic code.
- The answers returned to the user must be relevant and useful.
- Exceptions should be reported effectively to the user if they occur.
- When the question is of an unknown category or if the system is unable to find a relevant response to the question, general replies asking the user to rephrase the question should be returned.
- Questions of unknown categories should be stored in a separate 'machine learning' file which will be used periodically to add a higher level of functionality to the application.
- Any user should be able to test the application.

### **Business Rules**

- The administrators should be able to modify the knowledge base, the machine learning file, and administrative passwords.
- The users should be able to maintain records of any conversation or usage session they have with the application interface.

## DESIGN

Intello has been designed as a conversational bot, since the domain is limited to admission related queries and scalability is likely to be limited. AIML has been used as the underlying language for creating the knowledge base. The reasons for this design choice are explained in the content that follows. We then go on to explain the user interfaces, and then finally, the conceptual design (high level view).

### **Why a conversational platform using AIML is more appropriate for Intello?**

A chatbot built for a conversational platform such as one that replaces a “FAQ” is easier to implement using manual coding with AIML. The various points that prove this are:

1. The bot can store customised patterns depending on the developer. This feature can be extremely useful in a country like India, where the majority of the population is not aware of the basic rules of English grammar. For example, different users may ask a simple question such as “What is the admission procedure for BMS?” in different ways:

“How can I apply for BMS?”

“BMS admission process?”

“Admission procedure for bms?”

“How to take admission in bms?”

“Admition in bms how?”

2. So, various possibilities for spelling and grammar errors can be handled using customised patterns. Existing developer tools such as Api.ai generate contexts automatically based on the training provided by the developer, and may not cover a majority of the possibilities.

- Scalability is not a very important factor, since the number of possible queries regarding college related activities is likely to be limited. Scalability will be required occasionally, and to a small degree, which can be easily handled by the administrator.
- Every third party tool, even tools backed by industry giants such as Api.ai (Google) and wit.ai (Facebook) have limitations, as already mentioned. If we recognise these limitations in advance, we can make sure that they do not occur in the customised bot that we build.
- The engineering reason to build our bot simply is that we do not have to rely on the engineering decisions of someone else. The existing platforms satisfy the majority of the criteria needed by modern chatbots, but there are always some criteria that are specific to the chatbot being built and cannot be handled using existing tools.
- The business reason to build our own bot simply is that third party tools provide a limited number of queries per month, and during a busy period, the number of requests that our bot receives may surpass that. For example, api.ai provides 6000 queries a month. Moreover, third party tools can go down or get overloaded at any point of time, due to technical or business reasons.

- It is easier to track use cases that are not matched to any pattern by storing all such questions in a ‘machine learning’ file. Using the file, the knowledge base can be updated in a periodic and self-controlled manner.

## User Interfaces

As of now, we are operating Intello through the default IDLE (Integrated Development Interface for Python) interface. The general interface looks like:

```
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)]
on win32
```

Type "copyright", "credits" or "license()" for more information.

```
>>>
```

```
===== RESTART: C:\Python27\INTELLO\runme.py =====
```

```
Initializing AIML interpreter (please be patient)...
```

```
Loading std-startup.xml... done (0.59 seconds)
```

```
Loading standard\bot1.aiml... done (0.09 seconds)
```

```
Loading standard\bot2.aiml... done (0.04 seconds)
```

```
Loading standard\bot3.aiml... done (0.03 seconds)
```

```
Loading standard\bot4.aiml... done (0.02 seconds)
```

```
Loading standard\bot5.aiml... done (0.02 seconds)
```

```
Loading standard\bot6.aiml... done (0.01 seconds)
```

```
Loading standard\bot7.aiml... Loading standard\bot8.aiml... done (0.02 seconds)
```

```
Loading my-intello.aiml... done (0.00 seconds)
```

```
Interpreter Version Info: PyAIML 0.8.5
```

```
Kernel bootstrap completed in 1.06 seconds
```

```
>>> Hi
```

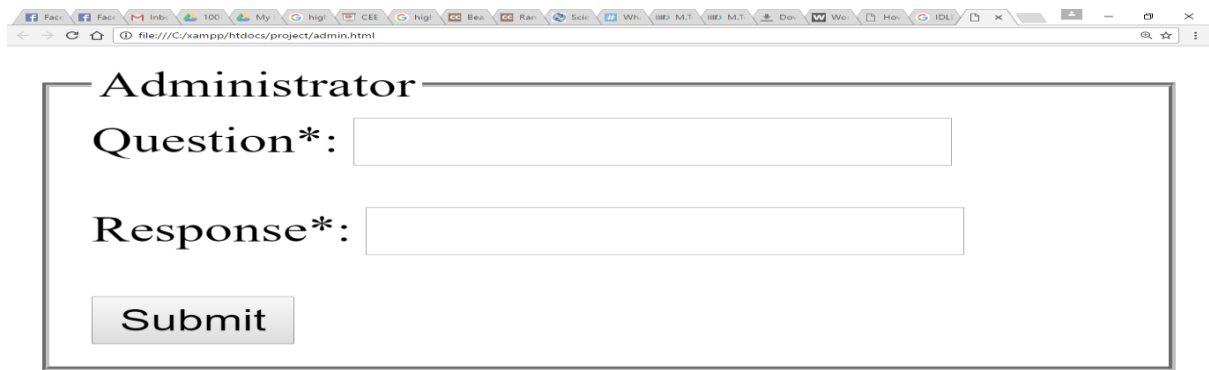
```
Hello, friend! How may I be of assistance?
```

```
>>>
```

```
“>>>” is used as the sign for the user to type his query.
```

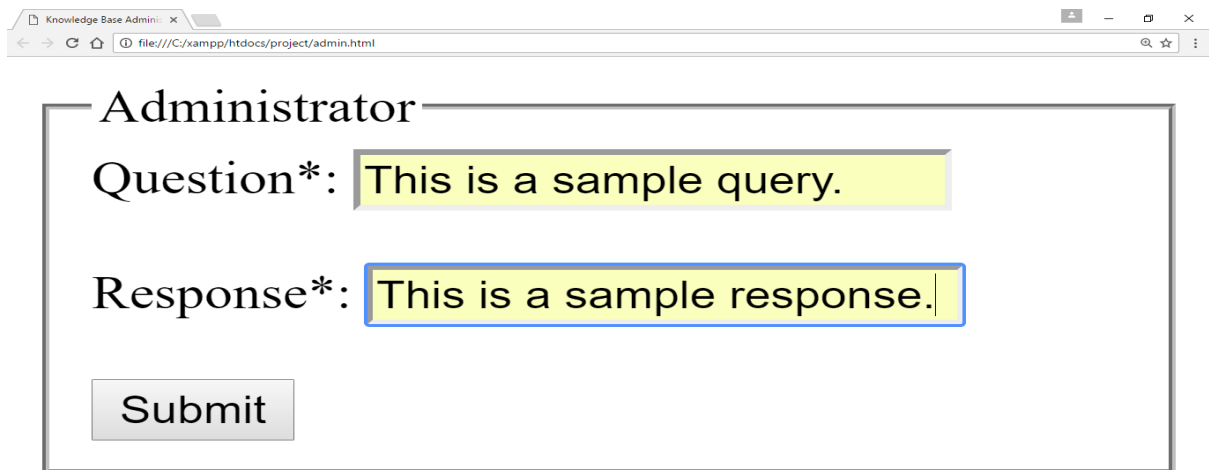
bot1, bot2, bot3, bot4, bot5, bot6 and bot7 are the AIML files that comprise the Knowledge Base of the system. Initially, the AIML interpreter is loaded, followed by the file ‘std-startup.xml’. Then, all the AIML files are loaded. It usually takes close to 1 second for the kernel to get loaded the first time it is started. Corresponding loads take less than 0.5 seconds on average.

Apart from the general chat interface shown above, Intello provides two more interfaces. One of them is used by the System administrator to update the knowledge base by updating an AIML file which is specifically used for storing answers to user queries received through user feedback (i.e., the queries that users couldn’t get answers to through the Intello chat interface). The admin is not supposed to understand how AIML works to update the knowledge base. All he needs to do is use the form interface shown below.



The screenshot shows a web browser window with the address bar displaying 'file:///C:/xampp/htdocs/project/admin.html'. The main content area is titled 'Administrator' and contains a form with two text input fields. The first field is labeled 'Question\*:' and the second is labeled 'Response\*:'. Below these fields is a 'Submit' button.

When the admin clicks the submit button, the AIML file gets updated. Now, the next time a user asks Intello the same question, he will get an appropriate response. The interface has been coded using HTML, and PHP has been used for backend processing. The PHP code updates the AIML file.



The screenshot shows the same web browser window as before, but now the input fields are filled. The 'Question\*' field contains the text 'This is a sample query.' and the 'Response\*' field contains the text 'This is a sample response.'. The 'Submit' button is still present.

The second interface is for user feedback. Sometimes, users may not get responses to every question that they ask. So, a special interface has been provided for them, through which they can tell us the question(s) that Intello couldn't answer. This way, Intello will keep evolving and be inclusive of user feedback. The interface (incorporated into the general Intello UI as a special query "submit a query") is shown below:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afaf1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Python27\INTELLO\runme.py =====
Initializing AIML interpreter (please be patient)...
Loading std-startup.xml... done (0.04 seconds)
Loading standard/AdmissionDomain\1.aiml... done (0.03 seconds)
Loading standard/AdmissionDomain\2.aiml... Loading standard/AdmissionDomain\3(i).aiml... done (0.01 seconds)
Loading standard/AdmissionDomain\3(ii).aiml... done (0.00 seconds)
Loading standard/AdmissionDomain\4.aiml... done (0.03 seconds)
Loading standard/AdmissionDomain\5.aiml... done (0.04 seconds)
Loading standard\intello-acronym.aiml... done (0.00 seconds)
Loading standard\intello-googlism.aiml... done (0.00 seconds)
Loading standard\intello-rhyme.aiml... done (0.00 seconds)
Loading standard\intello-submitquery.aiml... done (0.00 seconds)
Loading my-intello.aiml... done (0.00 seconds)
Interpreter Version Info: PyAIML 0.8.5
Kernel bootstrap completed in 0.24 seconds
>>> submit a query
PLEASE WRITE YOUR QUERIES BELOW!!!!.
>>> This is my query
YOUR QUERY HAS BEEN RECORDED. I WILL INFORM MY BOTMASTER KONAG ABOUT THE EVENT.
>>> |
```

When the user types in the query and presses enter, the “machine learning” file gets updated. This file stores all queries submitted by a user. The admin consults this file periodically and updates the knowledge base (using the admin interface) with answers to these questions.

The content of the machine learning file after the query is stored is shown below:

Current date & time 04/19/17 17:00:11

net password

Current date & time 04/22/17 23:48:55

store my query

Current date & time 04/26/17 00:42:56

What if I do not

Current date & time 04/26/17 00:43:34

This is my query

Note that “This is my query” was the entered query. The other queries stored above are from previous tests of this feature (not shown).

## Conceptual Design

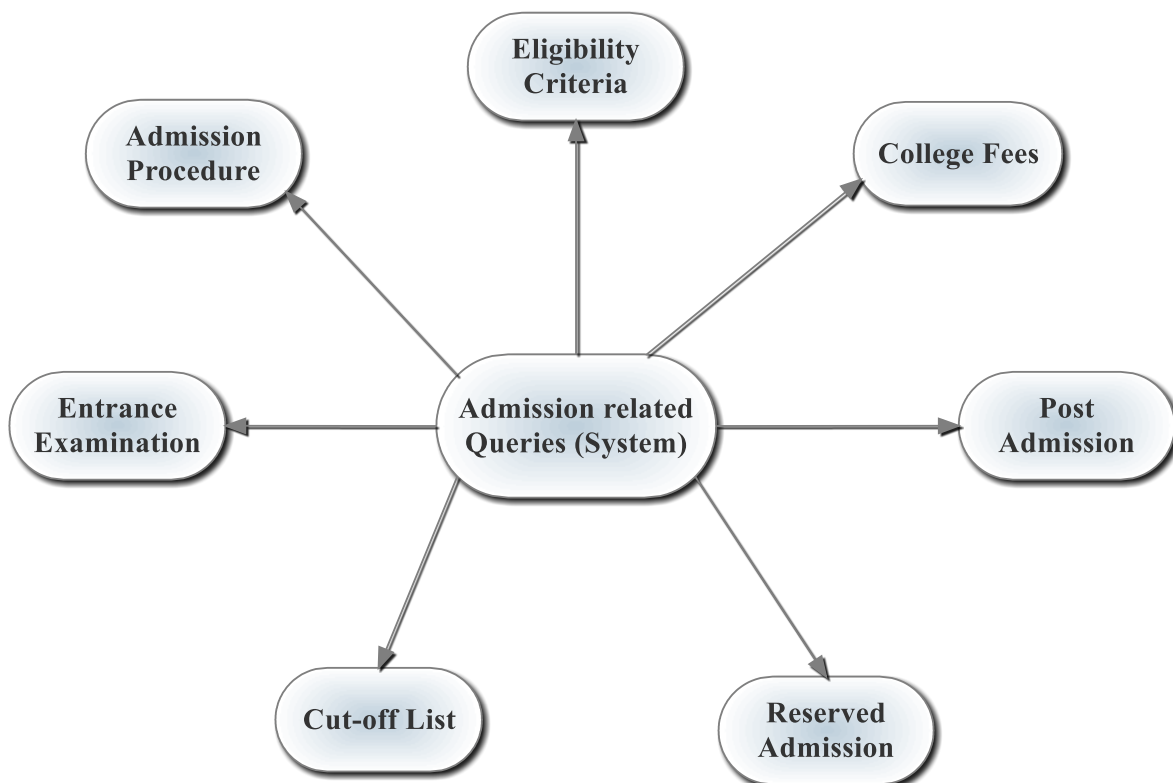
The system context diagram of Intello is shown in the following image:



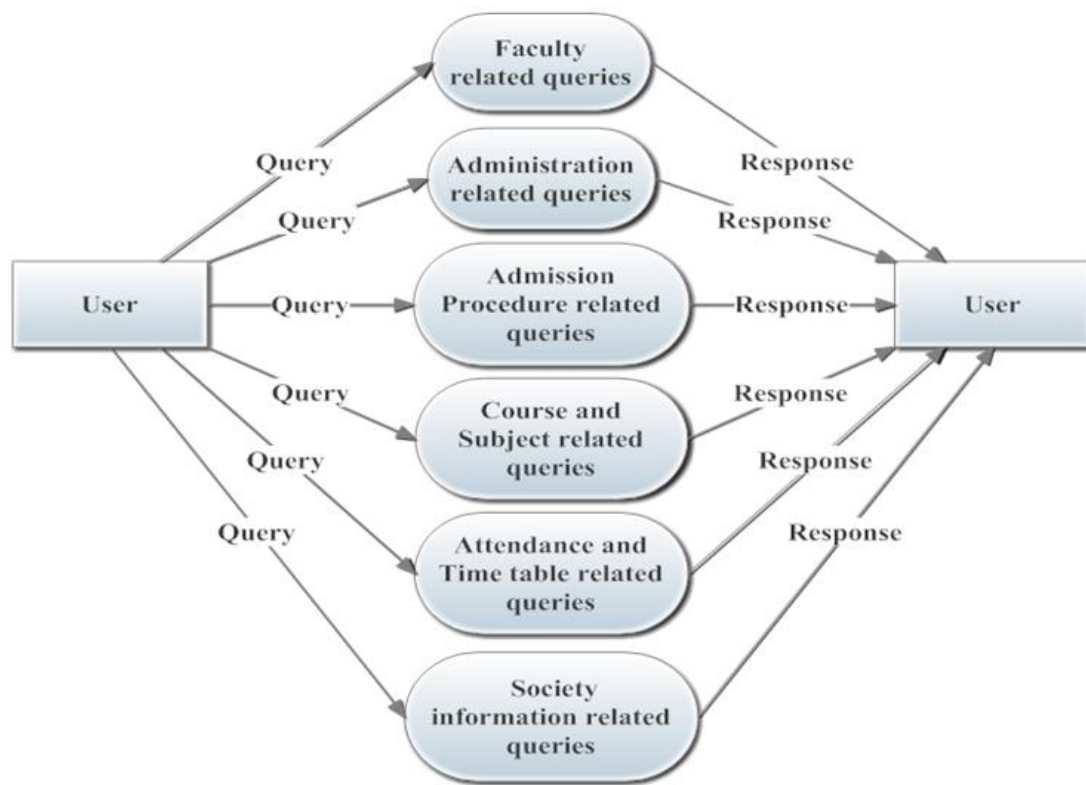
The user simply types in a query and the system generates an appropriate response to it by consulting its knowledge base. The response is then delivered to the user. Thus, the steps are:

- The user sends a query.
- The system consults its knowledge base for an appropriate response to the query (accomplished using pattern matching).
- Either an appropriate response is found, or a message saying that no response could be found.
- The system delivers the response or an equivalent message telling the user that a response could not be found. The response is delivered using both text and speech.

Currently, the Knowledge Base has been developed with answers to questions related to the “Admission Domain”. This domain can be further broken down and viewed as follows:



In the future, the Knowledge Base may be updated to contain queries related to many more domains. Here is a future model of the system:





## IMPLEMENTATION

The chatbot is created with the following components:

- Python Interpreter
- Brain File
- Pyttsx Python Library
- AIML Parser
- Pattern matching Algorithm

Python interpreter is foundation of this bot. It helps in carrying out the process and algorithm. The brain file built for this model is used as a repository such that the model does not need to be trained again and again. Pyttsx Python library is used to generate text-to-speech responses. AIML parser is used to parse the AIML files and check for syntax errors in the files. Pattern matching algorithm is used to generate responses by matching specific patterns to user queries.

### ➤ Procedure

Step 1: Create a brain file. It can be created by using AIML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<aiml version="1.0">
  <category>
    <pattern>HOW DOES ONE APPLY THROUGH THE ECA OR SPORTS QUOTA</pattern>
    <template>Check out the following link for the same:- http://www.du.ac.in/du/index.php?page=sports-eca-category</template>
  </category>
  <category>
    <pattern>WHAT ABOUT SPORTS QUOTA AND ECA ADMISSIONS</pattern>
    <template><srai>How does one apply through the ECA or Sports quota</srai></template>
  </category>
  <category>
    <pattern>SPORTS QUOTA AND ECA ADMISSIONS</pattern>
    <template><srai>How does one apply through the ECA or Sports quota</srai></template>
  </category>
```

Step 2: Load the brain file.

```

<!-- This category works with the Anna and Wallace AIML sets -->
<category>
<pattern>LOAD AIML A</pattern>
<template>
<!-- only allow this operation if the "secure" predicate is set to yes. -->
<condition name="secure">
    <li value="yes"><learn>aiml/*.aiml</learn></li>
    <li>You are not permitted to load AIML sets.</li>
</condition>
</template>
</category>

<!-- This category works with the Standard AIML Set -->
<category>
<pattern>LOAD AIML B</pattern>
<template>
<!-- only allow this operation if the "secure" predicate is set to yes. -->
<condition name="secure">
    <li value="yes">
        <!-- Load standard AIML set -->
        <learn>standard/std-*.aiml</learn>

        <!-- Load Admission Domain AIML set -->
        <learn>standard/AdmissionDomain/*.aiml</learn>

        <!-- Load any intello-specific AIML files -->
        <learn>standard/intello-*.aiml</learn>
    </li>
    <li>You are not permitted to load AIML sets.</li>
</condition>

```

Step 3: Wait for a request from a user or a client.

Step 4: User sends a request in text format.

- Query 1: Documents for admission.
- Query 2: please tell me about the college.
- Query 3: Where can i get the forms.
- Query 4: new semester starts on?

Step 5: The Model receives the request and forwards it to the pattern manager.

Step 6: Pattern matching algorithm is applied.

Pattern matching Algorithm

- Given<sup>[36]</sup>:
  - an input starting with word X, and<sup>[36]</sup>
  - a Nodemapper of the graph<sup>[36]</sup>:
- Does the Nodemapper contain the key \_? If so, search the subgraph rooted at the child node linked by \_.<sup>[36]</sup> Try all remaining suffixes of the input following X to see if one matches.<sup>[36]</sup> If no match was found, try<sup>[36]</sup>:
- Does the Nodemapper contain the key X? If so, search the subgraph rooted at the child node linked by X, using the tail of the input (the suffix of the input with X removed).<sup>[36]</sup> If no match was found, try<sup>[36]</sup>:
- Does the Nodemapper contain the key \*? If so, search the subgraph rooted at the child node linked by \*.<sup>[36]</sup> Try all remaining suffixes of the input

following X to see if one matches.<sup>[36]</sup> If no match was found, go back up the graph to the parent of this node, and put X back on the head of the input.<sup>[36]</sup>

- If the input is null (no more words) and the Nodemapper contains the <template> key, then a match was found.<sup>[36]</sup> Halt the search and return the matching node.<sup>[36]</sup>

Step 7: Use the brain to get an appropriate response

Response 1: You require the following documents for the admission procedure- 2 passport sized photographs, 10th class Mark sheet, 12th class Mark sheet, an ID proof (Aadhar Card, Electoral Photo Identity Card (EPIC), Permanent Account Number (PAN) Card, Passport), Character certificate and Migration certificate.

Response 2: You can get information about the college on the following link- <http://sscbsdu.ac.in/index.php/sscbs>

Response 3: Admission form is available on the University of Delhi's website. If you have already filled the online form, you don't need to fill any physical copy of admission form. For more information, please check the given link- [du.ac.in](http://du.ac.in)

Response 4: You can see Delhi University's academic calendar for the same.

Step 8: The response is forwarded to the user and converted into speech format using pyttsx.

```
import pyttsx

#Text-To-Speech
engine = pyttsx.init()
#rate = engine.getProperty('rate')
engine.setProperty('rate', 100)
while True:
    input = raw_input(">>> ")
    response = self.submit(input, self._sessionID)
    print response
    engine.say(response)
    engine.runAndWait()
```

## TESTING

**The Test Plan** – The chosen Test Plan is to test a few sub-domains and check whether:

1. The same question worded in different ways is answered appropriately.
2. Different questions which have similar words receive appropriate responses.
3. Whether keyword placement within a sentence affects the response.

Testing was carried out by a large sample of users on several sub-domains. It was discovered that the three criteria were met. The following examples (drawn from actual tests) illustrate this:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afaf, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Python27\INTELLO\runme.py =====
Initializing AIML interpreter (please be patient)...
Loading std-startup.xml... done (0.04 seconds)
Loading standard/AdmissionDomain\1.aiml... done (0.03 seconds)
Loading standard/AdmissionDomain\2.aiml... Loading standard/AdmissionDomain\3(i).aiml... done (0.03 seconds)
Loading standard/AdmissionDomain\3(ii).aiml... done (0.01 seconds)
Loading standard/AdmissionDomain\4.aiml... done (0.04 seconds)
Loading standard/AdmissionDomain\5.aiml... done (0.05 seconds)
Loading standard\intello-acronym.aiml... done (0.00 seconds)
Loading standard\intello-googlism.aiml... done (0.01 seconds)
Loading standard\intello-rhyme.aiml... done (0.00 seconds)
Loading standard\intello-submitquery.aiml... done (0.00 seconds)
Loading my-intello.aiml... done (0.00 seconds)
Interpreter Version Info: PyAIML 0.8.5
Kernel bootstrap completed in 0.41 seconds
>>> can you tell if there is any sports quota for admission into bms?
Check out the following link for the same:- http://www.du.ac.in/du/index.php?page=sports-eqa-category
>>> Is there any ECA quota?
Check out the following link for the same:- http://www.du.ac.in/du/index.php?page=sports-eqa-category
>>> Sports quota
Check out the following link for the same:- http://www.du.ac.in/du/index.php?page=sports-eqa-category
>>> |
```

The above screenshot illustrates that the question referring to Sports Quota or ECA admission was answered appropriately for 3 different versions of the question (Criterion 1)

The next example illustrates that two questions which are structured in a similar manner, have common keywords, but are representing different queries, are answered appropriately:

```

Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afaf, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Python27\INTELLO\runme.py =====
Initializing AIML interpreter (please be patient)...
Loading std-startup.xml... done (0.04 seconds)
Loading standard/AdmissionDomain\1.aiml... done (0.03 seconds)
Loading standard/AdmissionDomain\2.aiml... Loading standard/AdmissionDomain\3(i).aiml... done (0.02 seconds)
Loading standard/AdmissionDomain\3(ii).aiml... done (0.00 seconds)
Loading standard/AdmissionDomain\4.aiml... done (0.02 seconds)
Loading standard/AdmissionDomain\5.aiml... done (0.02 seconds)
Loading standard/intello-acronym.aiml... done (0.00 seconds)
Loading standard/intello-googlism.aiml... done (0.00 seconds)
Loading standard/intello-rhyme.aiml... done (0.00 seconds)
Loading standard/intello-submitquery.aiml... done (0.00 seconds)
Loading my-intello.aiml... done (0.00 seconds)
Interpreter Version Info: PyAIML 0.8.5
Kernel bootstrap completed in 0.22 seconds
>>> Tell me about the admission procedure
You can apply by following the procedure mentioned on the college website:- www.ssccbs.du.ac.in .
>>> Tell me about the documents required for the admission procedure
You require the following documents for the admission procedure- 2 passport sized photographs, 10th class Mark sheet, 12th class Mark sheet, an ID proof(Aadhar Card, Electoral Photo Identity Card (EPIC), Permanent Account Number (PAN) Card, Passport), Character certificate and Migration certificate.
>>> |

```

In the above screenshot, both questions start with “Tell me about” and end with “the admission procedure”. Moreover, “admission procedure” is a pattern in the knowledge base. A weaker pattern match algorithm would have failed to differentiate between the two queries. However, Intello manages to not just differentiate between the questions, but also delivers accurate responses to both (Criterion 2)

The next example illustrates that keyword placement within a sentence does not affect the response received as long as the question carries the same meaning.

```

Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afaf, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Python27\INTELLO\runme.py =====
Initializing AIML interpreter (please be patient)...
Loading std-startup.xml... done (0.02 seconds)
Loading standard/AdmissionDomain\1.aiml... done (0.03 seconds)
Loading standard/AdmissionDomain\2.aiml... Loading standard/AdmissionDomain\3(i).aiml... done (0.03 seconds)
Loading standard/AdmissionDomain\3(ii).aiml... done (0.01 seconds)
Loading standard/AdmissionDomain\4.aiml... done (0.04 seconds)
Loading standard/AdmissionDomain\5.aiml... done (0.05 seconds)
Loading standard/intello-acronym.aiml... done (0.00 seconds)
Loading standard/intello-googlism.aiml... done (0.00 seconds)
Loading standard/intello-rhyme.aiml... done (0.00 seconds)
Loading standard/intello-submitquery.aiml... done (0.00 seconds)
Loading my-intello.aiml... done (0.00 seconds)
Interpreter Version Info: PyAIML 0.8.5
Kernel bootstrap completed in 0.31 seconds
>>> What is the admission procedure?
You can apply by following the procedure mentioned on the college website:- www.ssccbs.du.ac.in .
>>> Admission procedure for BMS
You can apply by following the procedure mentioned on the college website:- www.ssccbs.du.ac.in .
>>> I want to know the admission procedure of BMS
Visit college website:-www.ssccbs.du.ac.in , to know more about the admission procedure.
>>> |

```

The pattern “admission procedure” is placed at the start of the question, the middle of the question, and the end of the question. The answer fetched in all the three cases is the same (Criterion 3).

Other tests carried out by users also revealed the above findings.

## RESULTS AND ANALYSIS

The final product matches the design constraints in terms of accuracy and diversity of questions.

The testing carried out by the chosen sample of users produced the following results:

Percentage of Queries answered: 75%

Percentage of accurate responses: 98%

Moreover, the average response time is negligible.

Every user who tested the bot claimed satisfaction with the responses, and approved further development.

Since the domain covered was “Admission Related Queries”, there were also some cross-referencing, i.e., fetching of a related, yet inexact response. This is something that the developers need to work on for better results overall.

For example, when a user typed in, “What about my documents”, the response fetched was related to submission of documents, when the user actually wanted to know about the retrieval of already submitted documents. This suggests that the pattern matching algorithm can be improved to work around the ambiguity that often comes packaged in a general user query.

## CONCLUSION

The final product that has resulted after development certainly matches up to the envisioned requirements. The accuracy of the responses justifies the whole idea that a chatbot can act as a worthy replacement for the FAQ section of the website of an educational institute. The ease of maintenance, usage, and the convenience offered to users are additional benefits that have been justified.

However, there's no denying that as more and more users test the application, more and more queries are discovered. Thus, maintenance is an essential feature, and would constantly require tech support. Also, as more and more domains get covered within the knowledge base and the size of the knowledge base grows, the responses may be slower. However, since the overall number of queries related to an educational institute is likely to be limited, this should not be a significant issue.

One thing that we did discover during the development of the AIML knowledge base is that there is no limit to human curiosity, and no matter how much one tries to cover all possible questions, the accuracy will never reach 100%. However, as the accuracy improves over time, we may reach some stable figure which is acceptable by any standard. 80% would be a good peak.

The increasing industrial demand for chatbots is understandable, and there is no denying how useful a chatbot can be for services like customer care, FAQs, and ease of navigation (on a website).



## REFERENCES

- 1) Ultimate Guide to Leveraging NLP & Machine Learning for your Chatbot - <https://chatbotslife.com/ultimate-guide-to-leveraging-nlp-machine-learning-for-your-chatbot-531ff2dd870c>
- 2) AIML Tutorial – <https://www.tutorialspoint.com/aiml/>
- 3) Chatbot - <https://en.wikipedia.org/wiki/Chatbot>
- 4) Building a Chatbot: analysis & limitations of modern platforms - <https://tryolabs.com/blog/2017/01/25/building-a-chatbot-analysis--limitations-of-modern-platforms/>
- 5) Features of Wit.ai vs Api.ai - <https://www.quora.com/Features-of-wit-ai-vs-API-ai>
- 6) AIML Based Voice Enabled Artificial Intelligent Chatterbot - International Journal of u- and e- Service, Science and Technology - Vol.8, No.2 (2015), pp.375-384 - <http://dx.doi.org/10.14257/ijunesst.2015.8.2.36>
- 7) IQABOT: A Chatbot-Based Interactive Question-Answering System – <https://smithware.files.wordpress.com/2010/11/final-report.pdf>
- 8) Implementation of ALICE chatbot as domain specific knowledge bot for BRAC U (FAQ bot) – <http://dSPACE.bracu.ac.bd/xmlui/bitstream/handle/10361/2390/Implementation%20of%20ALICE%20chatbot%20as%20domain%20specific%20knowledge%20bot%20for%20BRAC%20U%20%28FAQ%20bot%29.pdf.pdf?sequence=1&isAllowed=y>
- 9) Basic Tutorial on Artificial Intelligence Markup Language (AIML) - <http://www.instructables.com/id/BASIC-TUTORIAL-ON-ARTIFICIAL-INTELLIGENCE-MARKUP-L/>
- 10) Weizenbaum, J. ELIZA – A computer program for the study of natural language communication between man and machine. Communications of the ACM, 10(8):36–45 (1966).
- 11) Cliff, C. and Atwell, A. Leeds Unix knowledge expert: a domain-dependent expert system generated with domain-independent tools. BCS-SGES: British Computer Society Specialist Group on Expert Systems journal, 19:49–51 (1987).
- 12) Wallace, R., Tomabachi, H., and Aimles, D. Chatterbots go native: Considerations for an eco-system fostering the development of artificial life forms in a human world. Published (2003), online: <http://www.pandorabot.com/pandora/pics/chatterbotsgonative.doc>.
- 13) Bran, E. Chabot's in der Kundenkommunikation (Chabot's in customer communication). Springer, Berlin (2003).
- 14) Abu Shawer, B. and Atwell, A Using the corpus of spoken Afrikaans to generate an Afrikaans Chabot SALALS Journal: Southern African Linguistics and Applied Language Studies, 21:283–294 (2003).
- 15) Abu Shawer B. and Atwell A. Different measurement metrics to evaluate a Chabot system Proceedings of the NAACL'07 Workshop: Bridging the Gap: Academic and Industrial Research in Dialog Technologies. Pp.89-96, ACL(2007).
- 16) Abu Shawer B. and Atwal A. Machine Learning from dialogue corpora to generate chatbots. Expert Update journal, Vol. 6, No 3, pp 25-30 (2003).

- 17) Steven Birds, Ewan Klein and Edward Lapper. Natural Language Processing with Python by O'Reilly Media, Inc. (2009). [9] Allen Downey, Shroff Publisher "Think Python" (2012).
- 18) Python: <http://www.pandorabot.com/Pandora / pic / wallaceaimltutorial.html>"
- 19) David Sawyer McFernand, Publisher, O'Reilly "JavaScript and jQuery 2nd Edition The missing Manual" (2011).
- 20) IRS E-learning Glossary. (2005). [Online]. Available:  
<http://www.irs.gov/opportunity/procurement/article/0,,id=128685,00.html>.
- 21) A. Gracy, T. Butler, Beyond Knowledge Management: Introducing Learning Management Systems, Idea Group Publishing, (2006).
- 22) W. K. Harton, Designing Web-Based Training: How to Teach Anyone Anything Anywhere Anytime, John Wiley and Sons, (2001).
- 23) T. Murrari, "Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art", International Journal of Artificial Intelligence in Education, 10, 98-129,( 1998).
- 24) A.L.I.C.E. Artificial Intelligence Foundation. [Online]. Available:  
<http://www.alicebot.com/>
- 25) A. M M. Naves, I. Diniz, I., F.A. Barros, "Natural Communication via AIML Plus Chatterbot", V Symposium on Human Factors in Computers Systems (IHC 2002), Fortaleza – CE, 387, (2003).
- 26) A. M Turing, Computing machine and intelligence, (1950).
- 27) LOM (Learning Object Metadata). Draft Standard for Learning Objects Metadata. (2002) [Online]. Available:  
[http://ltsc.ieee.org/wg14/files/LOM\\_1384\\_12\\_1\\_v1\\_Final\\_Draft.pdf](http://ltsc.ieee.org/wg14/files/LOM_1384_12_1_v1_Final_Draft.pdf).
- 28) M. E. Batman, Intention, Plans, and Practical Reason, CSLI Publications, (1999).
- 29) Program D. Getting Started with Program D. (2005) [Online]. Available:  
<http://www.alicebot.org/resource/program/readme.html>.
- 30) R. Asaleeson, N. T. Schulte, Foundation of AJAX, Après,( 2005).
- 31) Fra Ling. [Online]. Available: <http://grraf.epsevg.upc.es/freeling/>.
- 32) ALICE BOT  
[https://en.wikipedia.org/wiki/Artificial\\_Linguistic\\_Internet\\_Computer\\_Entity](https://en.wikipedia.org/wiki/Artificial_Linguistic_Internet_Computer_Entity)
- 33) ALICE BOT- <http://www.alicebot.org/about.html>
- 34) Api.ai & Wit.ai- <https://www.themarketingtechnologist.co/api-ai-vs-wit-ai/>
- 35) Motion.ai - <http://www.veloxcore.com/chatbot-with-zero-coding-and-motion-ai/>
- 36) Pattern matching algorithm- [http://www.alicebot.org/TR/2001/WD-aiml-1.0-20010926-003.html#\\_Toc526430965](http://www.alicebot.org/TR/2001/WD-aiml-1.0-20010926-003.html#_Toc526430965)