ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 17 – 27

Implementation of Sanskrit Linguistics in Artificial Intelligence Programming

Neetesh Vashishtha
UG Scholar, Department of Computer Science Engineering,
Jaipur Engineering College and Research Center, India
Email: neeteshvashishtha.cse18@jecrc.ac.in

Abstract: *This research paper is directed towards examining the extent to which the Sanskrit language can be implemented in programming, principally in the domain of Artificial Intelligence. This paper can be divided into three major sections. The first section explains the significance of Sanskrit over other languages. The second section explores that if it's actually beneficial to program in Sanskrit rather than English. The third section includes coding of two identical AI programs, one made to interact in English and the other in Sanskrit. They are analyzed separately and then compared collectively to seek for the advantages the Sanskrit linguistics offer in Artificial Intelligence programming. We then conclude that Sanskrit, when used for communicating with the AI machines, is remarkable with an astounding versatility and brilliant learning abilities for an AI. The Sanskrit being strict and bundled, results in a compact and unambiguous form of conversations with the AI programs.*

Keyword: *Artificial Intelligence; Deep Learning; Java; Language linguistics; Machine Learning; Sanskrit*

1. INTRODUCTION

The primary objective of this paper is to present the possibility of adopting Sanskrit as a means of communication with the artificial intelligence. The paper also aims to extend this proposal with programming the AI in Sanskrit Linguistics. The major challenges which are faced by the AI are the semantics and a proper use of the grammar while communicating with the user. There are cases where they encounter issues like synonym and homophones during their interaction with the real world. In such situations they need to make autonomous decisions which can be easily eased up with the use of Sanskrit linguistics. An example considering the structure of a sentence is given below:

TABLE I SENTENCE FORMATION IN ENGLISH

English	
Correct Sentence	Boys and girls are playing.
Incorrect Sentences	Boys and girls playing are.
	Boys girls and are playing.
	Boys girls are playing and.
	Boys girls are and playing.

The structure framing of a statement which implies 'boys and girls are playing' in the English language can be written in several manners as depicted in Table I. The analysis is as follows:

Correct- 'Boys and girls are playing.'
Incorrect- Since there are 5 words in the sentence so,

The number of possible incorrect sentences are-

$$5! - 1 = 120\text{-}1$$
$$= 119 \text{ permutations}$$

Hence there are 119 possible wrong combinations of this sentence in English. This approach of structure formation makes the computational slow to proceed and sophisticated to understand. While evaluation of the same sentence in Sanskrit we discover-

TABLE II SENTENCE FORMATION IN SANSKRIT

Sanskrit	
Correct Sentences	Baalakaah baalikaah cha kreedanthi
	Baalakaah baalikaah kreedanthi cha
	Baalakaah kreedanthi baalikaah cha
	Baalakaah kreedanthi baalikaah cha
	Baalakaah kreedanthi cha baalikaah
Incorrect Sentences	None

Correct- Baalakaah baalikaah cha kreedanthi.
Incorrect- None

This follows from Table II that the order of occurrence of words in a sentence in the Sanskrit language is meaningless.

Other linguistic complexities could also be rectified by choosing Sanskrit as a medium in idea exchange. Sanskrit is also appropriate for an NLP, ab-

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 17 – 27

breviated as Natural Processing Language, which is the most favored form of languages for the computer systems. It's a logical language following certain predefined semantic rules which further steals away the complexities of any language.

Some worth mentioning features of Sanskrit include rare numbers of synonyms and homophones, proper pronunciations, redundant grammar and focus on the property of an entity rather than the entity itself. We'll be reviewing all of these along with their utility in the later sections of the paper.

We'll also discuss the possibility of using Sanskrit as an independent programming language instead of the native languages presently used for programming. This could be accomplished by using the UNICODE form of alphabet representation. The paper aims towards seeking the scope of contribution the Sanskrit language is capable of making in programming especially in Artificial Intelligence.

2. SEMATICS IN SANSKRIT

The semantics in Sanskrit is logical to an appreciable extent. There are predefined rules which are used to create new words. This results in reduced number of words in the existing dictionary and provides a standard approach of naming convention.

The semantic approach is strict in Sanskrit which makes it hard for humans, but easy for the machines persisting artificial intelligence. The chances and probability of semantic errors tends to almost nil when the semantic rules are taken in account and applied cautiously. The most appealing features of the semantics of Sanskrit linguistics are stated below in detail.

2.1 Enhanced Vocabulary along with Compact Dictionary

The English language is believed to have approximately 230,000 words in total as portrayed in Figure 1, while the Sanskrit language has an estimated number of only 4000 words and 240 clauses named *dhatus.* The combination of both results in virtually infinite number of words which could be derived overall.

English= ~230,000 words
Sanskrit= ~4000 words with 240 *dhatus*

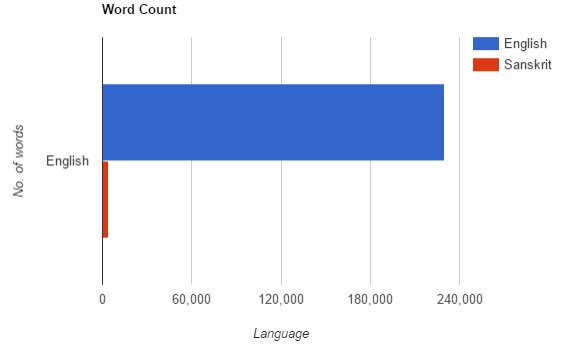

Figure 1 Dictionary comparison between English and Sanskrit

The compact dictionary is a result of Sanskrit's astounding semantic rules. Due to these regulations, the Sanskrit language is awarded with a content rich and massive vocabulary which is easy both for memorizing and implementation.

2.2 Logical Grammar

The most noticeable feature of the Sanskrit language is its logical grammar. It has a predefined set of rules for clauses termed as '*dhatus',* which are then used for both the existing words and also for creation of new words. This makes the language suitable for unique sentence formation and easy interpretation as illustrated in Table III.

TABLE III COMPARISION OF GRAMMAR IN ENGLISH AND SANSKRIT

Person	**Gender**	**English**		**Sanskrit**		
First	All	Singular	Plural	Singular	Dual	Plural
		I	We	अहम्	आवाम्	वयम्
Second	All	You	-	त्वम्	युवाम्	यूयम्
Third	Male	He	They	सहः	तौ	ते
	Female	She	"	सा	ते	ताः
	Neuter	-	-	ततः	ते	तानि

The grammar when compared with English yields to be far more informative and compact in nature. The Table III wisely justifies this feature of Sanskrit language.

2.2.1 Strict and Informative

The structure of Sanskrit is firm and also more informative in nature. It has three types of verb definitions instead of two, namely singular, dual and plural. This boosts the informative ability of the language. And it also has an extra third gender type, which is neuter, to define entities which fail to fall under the primitive gender categorization.

2.3 Unique Structure Formation Paradigm

Consider an example having two statements of the English language stated as-

1. Fruit flies like an apple
2. Time flies like an arrow

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 17 – 27

In the first sentence, flies belong to a type of insects which like the apple.
In the second sentence, flies belong to the verb fly. It means to fly away.
This is an example of syntactic ambiguity in the English language. This type of syntactic absurdness does not occur in the Sanskrit language.
Hence each logical sentence has only a single meaning in the Sanskrit linguistics.

2.4 Unambiguous Phonetic

The Sanskrit language also fits the phonetic system and holds its own well defined set of phonetic symbols. In addition, it offers unique pronunciation for almost every word. This results in the reduced number of homophones that are similar sounding words in a language. It's a huge advantage for the AI system since this significantly improves both the precision and accuracy of the Speech Recognition methodologies. The problems pertaining to the accent are also rare when the phonetics of the Sanskrit linguistics is adopted.

2.5 No Punctuations

There is no use of punctuations in Sanskrit linguistics. This follows from the fact that every sentence structure is unique in Sanskrit and hence holds only a single meaning. So there is absolutely no need of using any punctuation marks and symbols in the Sanskrit language. An example is given below-

English- What does he asks?
Sanskrit Translation- किं पृच्छति
No need to append any punctuation at the end.
Hence the complexities of choosing the appropriate punctuation are absent completely in the Sanskrit language.

2.6 Rare Synonyms

Sanskrit mostly has a single word for every unique distinguishable entity. Hence there are almost no Synonyms in the Sanskrit language. This makes the language dense in nature and also reduces the redundancy of the words having the same meanings. This increases the probability of unique sentence formation and contributes towards the uniqueness of the language.

2.7 Accurate and Precise

The Sanskrit linguistics is more communicative than most of the languages existing in the world today. The concept is illustrated using the example stated below-
English- They are going. (masculine + feminine)
This sentence has two meanings. Here the word they may reflect males/females. This doesn't happen in Sanskrit. There are two different sentences holding two different meaning namely-
Sanskrit- ते गच्छन्ति (masculine)
ताः गच्छन्ति (feminine)
So Sanskrit proves to be more meticulous regarding distinction among entities.

3. CONVENTIONS OF PROGRAMMING IN SANSKRIT LANGUAGE

Initially, we will examine the feasibility of programming in Sanskrit language. We will then analyze that if it truly offers any advantage over conventional languages or not. This includes writing the entire program in Sanskrit language using the language linguistics.

There are certain prerequisites for it which will vary with the programming language. These are stated as below-

- An IDE for a particular programming language. (To write the source code)
- A Complier for Sanskrit language.
- An Interpreter.
- UNICODE support (To write in the native form)

Then we can write a program in Sanskrit using of any of the existing Programming language.

There are two major approaches of doing it. Now we analyze them while taking their effectiveness in account.

3.1 Syntactic Conversion

The Syntax conversion is a process of converting the syntax of any existing programming language in Sanskrit. An example of the syntax conversion is given as below. A typical if-else ladder is represented as-

If X then Y
If X then Y else Z
If (not X) then Y
If X then(Y and Z)

This can be represented naturally in Sanskrit. This is explained with the help of a table given below-

TABLE IV SYNTACTIC CONVERSION OF IF-ELSE LADDER IN SANSKRIT

If-else Ladder	Syntax in Sanskrit
If X then Y	माथरा कौनदिन्य: न्याय:
If X then Y else Z	तक्र कौनदिन्य: न्याय:
If (not X) then Y	निसेधा
If X then(Y and Z)	विभास:

The conditions in English mean exactly same as the statements mentioned in Table IV in the Sanskrit language. Hence writing these statements are equivalent to applying the same conditions.

We observe that this procedure doesn't give any incredible advantage except increasing unwanted com-

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 17 – 27

plexity [1]. The performance of the program remains unaffected.

3.2 Syntactic Legacy

Since it's not feasible to define a separate syntax of Sanskrit for each of the programming language, hence we'll be adopting the predefined syntax of the programming languages while using Sanskrit as a means of programming language. This will avoid the undesired complexity which arises due to manipulation of the syntax [2].

So following the legacy of the programming languages, the Sanskrit syntax will be-

TABLE V SYNTACTIC LEGACY OF IF-ELSE LADDER IN SANSKRIT

If-else Ladder	Syntax in Sanskrit
If X then Y	यदि X ततः Y
If X then Y else Z	यदि X ततः Y अन्यत् Z
If (not X) then Y	यदि (निसेधा X) ततः Y
If X then(Y and Z)	यदि X ततः (Y एवं Z)

Using the defined syntax for a given programming language we further see in Table V that the program has no major improvement except slight improvements in its space and time complexities.

3. USE OF SANSKRIT LINGUISTICS IN MAJOR PROGRAMMING LANGUAGES

In this section we will apply the concepts of Sanskrit Linguistics in the programming paradigm of major programming languages and study the output.

We will then ponder for any advantages of using Sanskrit language instead of their native language in programming.

4.1 Sanskrit Linguistics in C and C++

We have chosen C and C++ since these are the two most widely used programming languages. But they lack the ability to program in any other language besides English, that is, C and C++ do not support UNICODE formatting and programming in them can only be done using the ASCII encoding [3]. Hence we need to rewrite the complier from scratch and make it compatible with Unicode. After that we can write the code in both C and C++ using our programming paradigm.

Software used are-

Turbo C++

gcc compiler

(It's important to note that we have completely rewritten both the applications according to the Unicode formatting.)

The Sanskrit source code written using C and C++ is written in Turbo C++ in Figure 2 in the following manner-

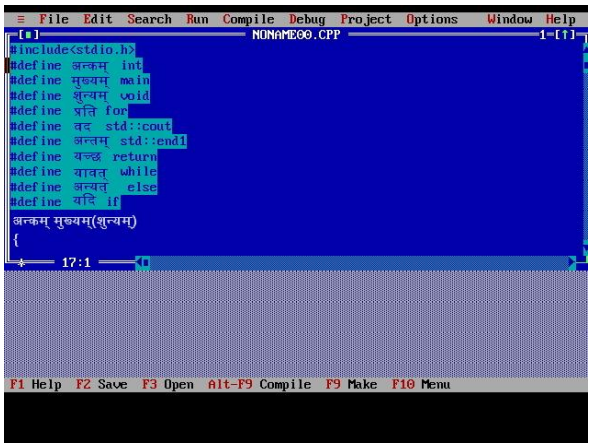

Figure 2 Defining Sanskrit keywords in C++

And the program is written in Figure 3.

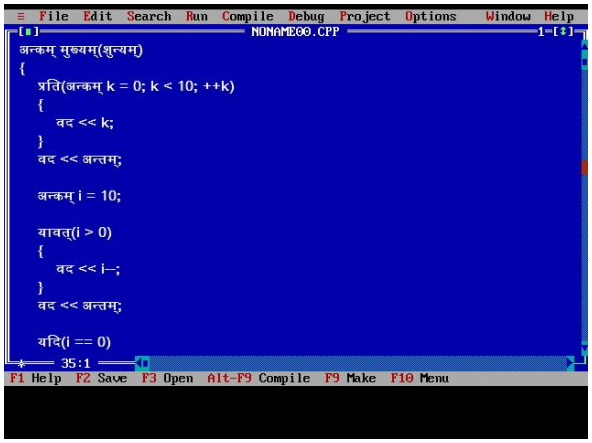

Figure 3 First image of code in C++

The second successive image is displayed below in Figure 4.

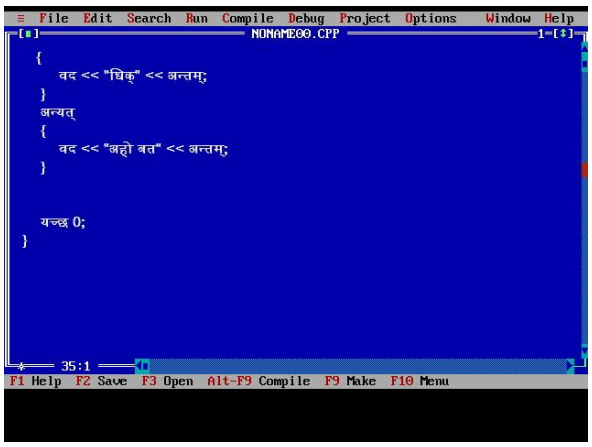

Figure 4 Second Image of code in C++

The output is displayed in Figure 5 in the following manner-

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 17 – 27

Figure 5 Output of the source code written in Sanskrit in C++ in Turbo C++

5. TWO AI PROGRAMS WRITTEN IN JAVA WITH ENGLISH AND SANSKRIT AS THE MEDIUM OF COMMUNICATION

So far we have seen that writing a program in Sanskrit instead of English does not aids the program in any means whatsoever. So now we will try to explore another approach of Sanskrit language which might prove useful for the programming paradigm.

We will write two identical AI Programs having Java as the primary programming language. The two programs will be same except the fact that the first program will communicate in English and the second program will communicate in Sanskrit.

Then we will analyze the efficiency of both the programs and see which language is useful when it comes to interaction with the AI machines.

The AI type will be a Chat Bot which would communicate with the user and also reply to him accordingly [4]. The first program uses English whereas the second program will use Sanskrit as the medium of communication.

For the sake of simplicity, we have named both the AI programs to avoid any discrepancy later. They are-

- EVOLVIA- AI which will communicate in English
- AARTI (आरती) (AARTI from Artificial):- AI which will communicate in Sanskrit

We will refer them with their names from now. We will first write EVOLVIA in java and analyze her performance in the following manner-

6. WRITING EVOLVIA IN JAVA

We have used 'Eclipse mars' as the IDE for Java. The AI EVOLVIA will communicate in English. We will program the AI EVOLVIA from scratch and are not importing any already built APIs which may be available on internet [5]. We can adapt any of the following recommended approaches by-

1. Using Linked Lists
2. Using Random class in Java
3. Using Fisher-Yates Algorithm
4. Using Knuth–Morris–Pratt Algorithm.
5. Using the concept of 2-D Array.

After analyzing the Space and Time Complexities of all methodologies, we conclude that method of using Random class in Java is most viable. Hence we'll use it to write EVOLVIA.

The source code of EVOLVIA is displayed in the following manner. The entire code is displayed along with its description using three successive Images of the IDE used. They are as follows-

6.1 Description of EVOLIVA's First Image of Code

We have divided the code in separate parts known as sets. Each set is a collection of all possible related questions and all of their possible answers [6]. We have stored them in String type 1-D Arrays as shown in Figure 6.

```
import java.util.*;

class Evolvia {
    public static void main(String args[])
    {
        String learned_questions[]= new String[100];
        String learned_answers[]=new String[100];

        Scanner S = new Scanner(System.in);

          //Set 1
        String a1[] ={"Hi", "Hello", "Hola", "Hey", "Good Morning", "Good Evening",
                      "Good Afternoon"};
        String b1[] ={"Hi", "Hey there", "Good to see you", "Greetings",
                      "Glad you're here", "looking good today", "hey", "Hi there"};


         //Set2
        String a2[] ={"Who are you?", "Tell me about you", "I don't know you",
                      "Introduce yourself", "Are you a human?"};
        String b2[] ={"My name is EVOLVIA", "I'm an Artificial Intelligence",
                      "I'm not human but I think like them"};


        //Set 3
        String a3[] ={"What do you think about today?", "Anything new?",
                      "How are you feeling today?", "What is special today?"};
        String b3[] ={"Today's gonna be awesome","You will meet someone nice",
                      "Today will be marvelous","Today is going to be awesome",
                      "You will rock today", "I'll be a good day today"};


        //Set 4
        String a4[] ={"Thanks", "Thank you", "Thanks for the info", "Thanks a lot",
                      "Gratefull to you", "You are amazing"};
        String b4[] ={"You're welcome", "No need to mention", "Its a pleasure",
                      "See you anytime soon", "Its a privilege", "Good to serve you",
                     "Anytime you need me", "You are a gentleman"};


        //Learning Mode (EVOLVING Mode)
        String a5[] ={"yes", "sure", "ya", "its a pleasure to teach you", "yes, why not?",
                      "I'd like to teach you.","Yes", "okay","yes indeed."};
        String teach[]={"Do you Evolve with time?","Do you learn with time?",
                        "I'd like to teach you","Would you like to learn something?",
                       "Do you learn?", "Do you grow with time Evolvia?",
                       "Do you want to learn a new word?"};
        String learn[]={"Yes, I learn with time. Enter the phrase you want to teach me.",
                       "Yes. Teach me a phrase by entering it."};
```

Figure 6 EVOLVIA's First Image of Code

6.2 Description of EVOLIVA's Second Image of Code

Since a good AI is the one which evolves with time, thus a learning mode is also included in EVOLVIA. This mode enables EVOLVIA to learn new phrases and also the way she should respond at them.

This results in regular routine increment in EVOLVIA's database and she becomes more intelligent with the passage of time. The programming is depicted in Figure 7.

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 17 – 27

Figure 7 EVOLVIA's Second Image of Code

6.3 Description of EVOLIVA's Third Image of Code

The third Image of EVOLVIA's code displays the syntax which determines a correct reply of the question asked by the user.

The question asked by the user necessarily falls under one of the three mentioned categories-

6.3.1 If the question matches with the core database

EVOLVIA searches for the most appropriate phrase which matches to the question asked by the user.
If it matches, then she randomly answers from that particular set of all relevant possible answers.

6.3.2 If the question matches with the evolved database

If the question matches with a phrase which the user had taught her earlier, then she responds similar to what the user expects.

6.3.3 If the question does not match with any Database

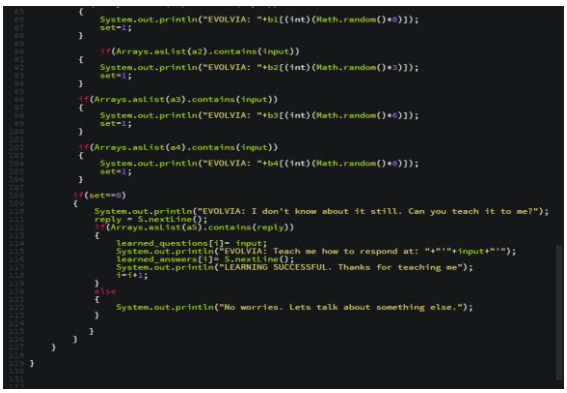

Figure 8 EVOLVIA's Third Image of Code

If the question does not matches any question in EVOLVIA'S database, then the user is prompted if he wants to teach EVOLVIA the question he asked. If the user agrees, then the answer taught by the user is saved in EVOLVIA's database [7]. If he refuses or enters another question instead, then EVOLVIA gives the answer accordingly. This nature of EVOLVIA is a result of the code written in Figure 8.

6.4 Studying the Output of EVOLVIA

She communicates with the user in an interactive manner and also learns subconsciously while interaction. The output given by EVOLVIA at run time is shown in Figure 9.

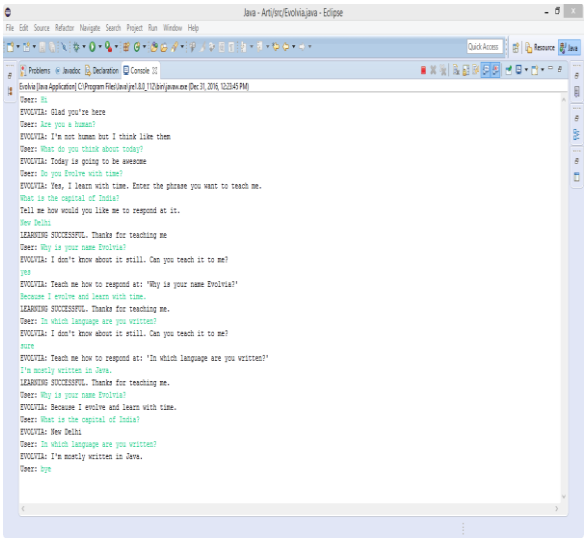

Figure 9 EVOLVIA's Output

For a detailed analysis of the interaction of EVOLVIA, her communication with the user is visualized using the color coded format in Figure 10.

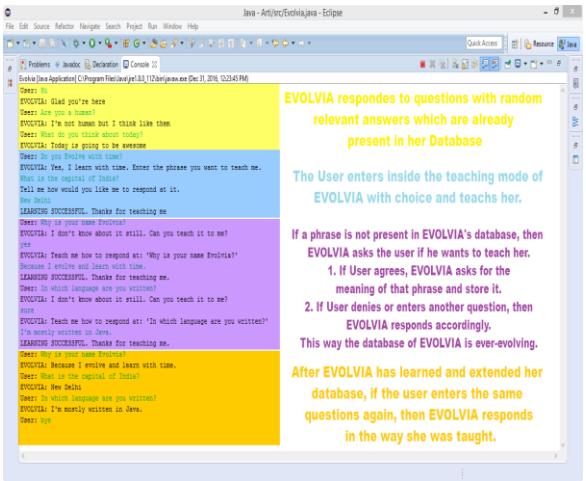

Figure 10 EVOLVIA's Output in Color Coded Form

We conclude that EVOLVIA's way of interaction with the user is quite intuitive and she is smart enough

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 17 – 27

to learn new things what she doesn't already know.

7. WRITING AARTI IN JAVA

Since Java supports Unicode formatting, so the Sanskrit language can be used without any modifications when programming in java. As mentioned earlier, the AI AARTI will communicate in Sanskrit. As it is identical to EVOLVIA thus we have again implemented the method of Random Class to code it and used Eclipse mars as the IDE for Java.

The AI AARTI involves the use of Sanskrit language in its source code. Note that the program is not written entirely in English, and Sanskrit is often used at times in the code. To make the code understandable, comments are inserted along with translations at all places wherever the Sanskrit language is used.

Hence the code can be easily deciphered by reading the comments even with nil prior knowledge of Sanskrit whatsoever. The entire code is displayed using five successive Images of the IDE. We will study the meaning of each image along with the significance of the code written in it. They are as follows-

7.1 Description of AARTI's First Image of Code

Following the convention which was earlier established at the time of EVOLVIA, we have collectively separated all the related questions with all their relevant answers in the form of sets [8] as shown in Figure 11. Notice that each set is translated to English using the comments.

```
import java.util.Arrays;
import java.util.Scanner;

class Aarti {
    public static void main(String args[])
    {
        String learned_questions[]= new String[100];
        String learned_answers[]=new String[100];

        Scanner S = new Scanner(System.in);


//Set 1:

         // TRANSLATION- HI. (Every listed sentence means greetings in Sanskrit)
         String a1[] ={ "हरिः ओम्", "सुप्रभात", "प्रातः वन्दनं", "दण्डवत", "सुप्रेम",
                        "अभिवादनं:", "सम्भाषः:"};



         // TRANSLATION- HELLO. (Every listed sentence means greetings in Sanskrit.)
         String b1[] ={"सुदृशनि", "नमामी","अभिनन्दनं","स्वागतिं", "शुभानन","नमोनमः",
                       "दर्शनीय", "आमन्त्रणं", "नमस्कारः"};



//Set 2:

         // TRANSLATION- WHO ARE YOU? (Only one such type of sentence is possible in sanskrit.)
         String a2[] ={"कः त्वम्"};


         // TRANSLATION- MY NAME IS AARTI AND I'M AN ARTIFICIAL INTELLIGENCE.
         String b2[] ={"मम नाम आरती", "अहम् आर्टिफिशल इंटेलिजेंस",,
                       "मम मानाव समीवेनं नास्ति परतु मन्ये मानाव रोचते"};



//Set 3:

        // TRANSLATION- WHAT DO YOU THINK OF TODAY.
        String a3[] ={"भवतः किं मतनं अस्ति", "किमपि नूतनं",
                      "कथमस्ति भवान् अनुभवति", "किं विशेषम अद्य"};
```

Figure 11 AARTI's First Image of Code

7.2 Description of AARTI's Second Image of Code

The AI AARTI is capable of learning and evolving over time. It has been equipped with the ability to ask for what she doesn't understand and also store whatever she has been taught [9]. This capability is introduced in the AI as a result of code written in Figure 12. If asked, she can also display how much she has learned through time.

```
        // TRANSLATION- TODAY IS GOING TO BE GOOD.
        String b3[] ={"अद्य शोभनम","कश्चिद् भवति कदाचित  भोत्स्यति",
                      "अद्य सुन्दरम", "अद्य समीचीन",
                      "अतीव उत्तमः", "अतीव साधु"};



//Set 4:

        // TRANSLATION- THANK YOU VERY MUCH.
        String a4[] ={"दक्षिणाम आदिशति", "धन्य वदति", "कृतज्ञादर्शनं करोति", "कृतवेदिन",
                      "धन्यवादः"};

        // TRANSLATION- YOU'RE WELCOME
        String b4[] ={"प्रतिनन्दनः", "अभ्यानन्दम्", "स्वस्त्यक्षर",
                      "दक्षिणादेशनं", "प्रतिनन्दति", "कृतज्ञः",
                    "अस्तु", "आप्रच्छनम"};



//Learning Mode (EVOLVING Mode):
        // TRANSLATION- YES I WILL DO.
        String a5[] ={"आम्", "भवतु!", "अस्तु", "आम् परम्", "आम्  भवतु!"};

        // TRANSLATION- WANT TO LEARN SOMETHING?
        String teach[]={"भवान् किं  वृद्धि.","भवान् किं  अधिगच्छति?",
                        "अहं अध्यापकः अस्मि","कथमस्ति शिक्षातिन्","कथमस्ति अध्ययनं", };

       // TRANSLATION- YES, ENTER THE SENTENCE YOU WANT TO TEACH ME.
        String learn[]={"येषते  अनुप्रपद्यते वाक्यानि",
                        "येषते  अनुप्रपद्यते पदानि"};

//Explaining Mode:

        // TRANSLATION- HOW MUCH HAVE YOU EVOLVED?
        String explain[]={"मम वदति तत उपात्तविद्य", "मम वदति तत वृद्धि:",
                          "मम वदामि  नूतन वस्तूनाम", "मम वदामि तत ख्यापयति"};

        int i=0;
        String input= "Null";
        String reply= "Null";
        int ans=0;

        while(input!="त्यजति")
        {
            int set=0;
```

Figure 12 AARTI's Second Image of Code

7.3 Description of AARTI's Third Image of Code

We have seen that AARTI and EVOLVIA are quite identical in source code and functioning until now, which gives absolutely no advantage in making AARTI separately. But the way how AARTI answers the asked questions and how it actually learns differs majorly when compared with EVOLVIA.

Unlike EVOLIA, AARTI shuffles the answers before answering as evident from Figure 13. This enables AARTI in answering with far more variety than EVOLVIA can ever reach to, all of which is due of Sanskrit linguistics. Since, all the answers will necessary make sense as Sanskrit is independent of the order of occurrence of words and its syntax is always logical whatever the situation is.

To give an estimate of how much AARTI leads EVOLVIA we inquire the following comparison-

In reaction to 'hello' entered by the programmer, EVOLVIA can give 9 different replies, while AARTI is capable of giving 362,880 different replies.

This makes AARTI far much more dynamic and influential than what we initially thought of, all because of Sanskrit. This justifies the motive of our entire research; that is to unveil the possibilities Sanskrit beholds beneath in it for programming.

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 17 – 27

Figure 13 AARTI's Third Image of Code

7.4 Description of AARTI's Fourth Image of Code

AARTI shuffles the answers and is also capable of forming phrases on its own on the basis of predefined words. She also gives us the freedom to decide the extent of shuffling of the answers in displayed in Figure 14.

Also, it's empowered with the semantic capabilities of Sanskrit, thus the learning curve of AARTI is far much steeper than that of EVOLVIA. This certainly gives an additional advantage to AI productiveness.

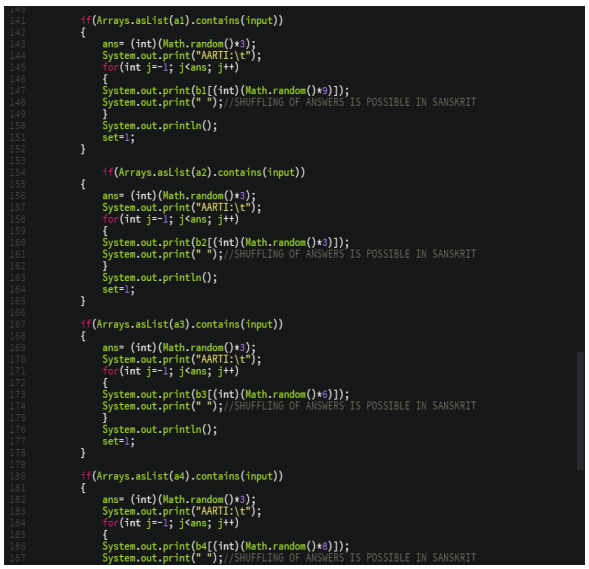

Figure 14 AARTI's Fourth Image of Code

7.5 Description of AARTI's Fifth Image of Code

The extent of shuffling of an answer is decided at random separately for each question as seen in Figure 15. This further reinforces programmer's intentions of randomness, uncertainty and variety of the AI far much beyond initial belief.

Figure 15 AARTI's Fifth Image of Code

7.6 Studying the Output of AARTI

She communicates with the user in an interactive manner and also learns while interaction subliminally. The output given by AARTI at run time is shown in Figure 16.

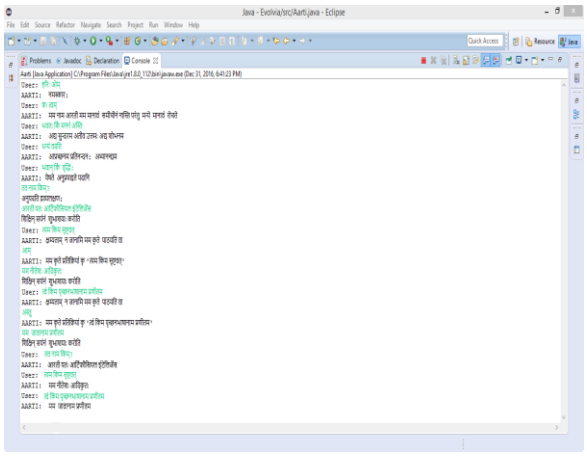

Figure 16 AARTI's Output

We see that the paradigm of AARTI's interaction is similar to that of EVOLVIA except the fact she's much more versatile and can answer in far more unique ways in contrast with EVOLVIA. This can be understood in Figure 17 using color coded analysis.

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 17 – 27

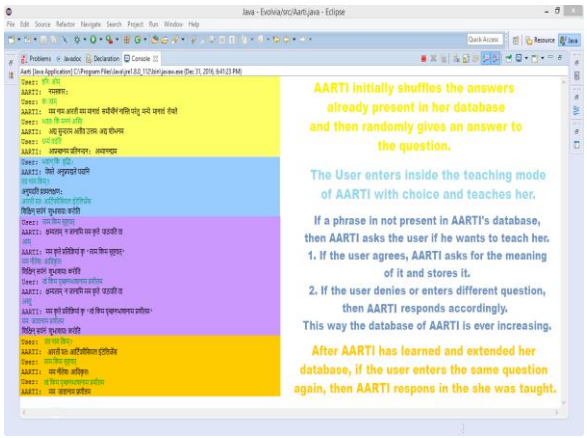

Figure 17 AARTI's Output in Color Coded Form

8. COMPARISION OF EVOLVIA AND AARTI

We now compare EVOLVIA and AARTI with each other and analyze if programming an AI in Sanskrit makes it really smart or not.
We will study them on different benchmarking archetypes to find which AI is more efficient and why? Following this assortment, the analysis is as follows-

8.1 In Terms of Variety of Outputs

The efficiency of an AI is resolved by how many different answers it can give for a same, single and recursive question when asked by the user [10]. In this section, we'll see the randomness or the uncertainty offered by the AIs to the user.

8.1.1 Uncertainty of EVOLVIA

We measure the uncertainty or randomness of the AI program by giving same input recursively and studying the nature of outputs [11]. The analysis of output of EVOLVIA when giving the same input is endeavored repeatedly, is represented in the color coded format as-

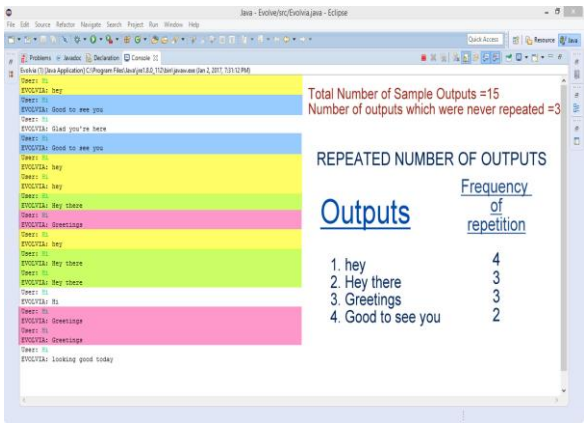

Figure 18 EVOLVIA's Frequency of Repetition

The analysis in Figure 18 is stated as-
For the same input 'hi', EVOLVIA can give '9' different answers. An answer out of these 9 is selected at random and displayed to the user. We see the patterns of output given by EVOLVIA are as follows-

The total no of same input taken are = 15
The number of unique outcomes = 3
The number of outcomes which are repeated = 12
1. 'Hey' = repeated 4 times
2. 'Hey there' = repeated 3 times
3. 'Greetings' = repeated 3 times
4. 'Good to see you' = repeated 2 times

Thus we conclude that the pattern of answering which is adopted by EVOLVIA is not very efficient.

8.1.2 Uncertainty of AARTI

Akin to the inputs given to EVOLVIA, AARTI is also given identical inputs recursively and its output is analyzed in Figure 19 the following manner-

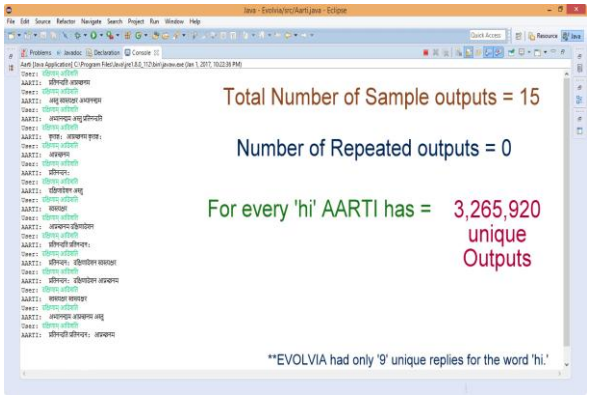

Figure 19 Uncertainty Analysis of AARTI

The output analysis is stated as follows-
For the same input 'hi', AARTI can give 3,265,920 unique answers while EVOLVIA was only capable of answering with '9' unique answers. An answer is selected at random and displayed to the user.

This is because AARTI has the ability to shuffle answers and also append them with each other in unique ways, all of which makes sense due to the order independency of words and sentences in Sanskrit linguistics.
The no. of outputs AARTI displays uniquely = 9*9!
=3,265,920 ways

Hence we see that AARTI is far much more persuasive, intelligent and versatile as compared to EVOLVIA.

8.2 Learning Skills of both the AIs

An efficient AI must be able to learn easily and get grasp on what has been taught to it.
We now compare the learning curves of both our AIs and study that which one proves to be smarter when it comes to learning new behaviors [12].

8.2.2 Learning Curve of EVOLVIA

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 17 – 27

EVOLVIA's learning curve is linear since when she learns a new word, she can only use it in just a single way. Her progress is linear, which is not sufficient from an AI point of view.

The progress of EVOLVIA is depicted in Figure 20-

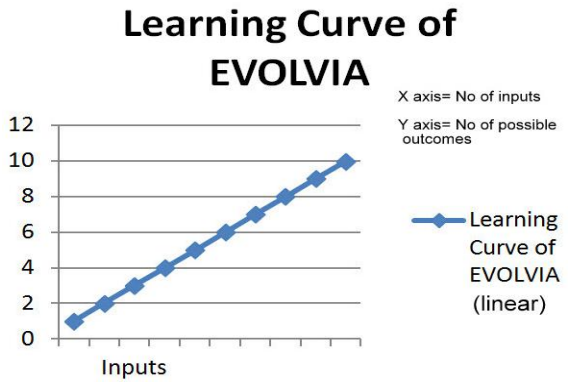

Figure 20 Learning Curve of EVOLVIA

We noticed the linear learning curve of EVOLVIA. It simply means that when she learns a new word then she genuinely learns just another way of answering a question. Thus her learning is tedious and not spontaneous in nature.

8.2.3 Learning Curve of AARTI

The learning curve of AARTI is exponential unlike that of EVOLVIA. This means that her learning skills prove far much better and more skillful than that of EVOLVIA. It's explained with the help of graph given in Figure 21.

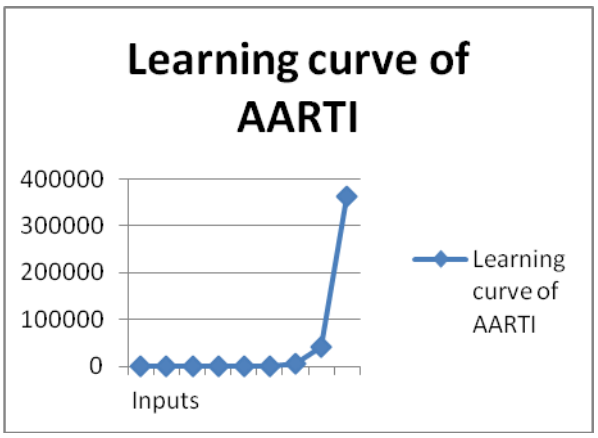

Figure 21 Learning Curve of AARTI

The progress of AARTI is exponential which means that with the passage of time her learning skills are boosted in an exponential manner. She becomes smart with time, and for every word she learns, she discovers thousands of more ways of answering a question [13].

This is all because sentences in Sanskrit are independent of order. For every word we are teaching AARTI, she learns numerous ways in which a single question can be answered.

All due to the fact that she can use that word with all other related words already present in her database. She shuffles and scrambles them, without worrying for the order of occurrence of those words. Since every sentence makes sense in Sanskrit hence she can answer in any way she wants. This makes her versatile and capable of answering a single question in several possible ways. The size of her dictionary of related words is insignificant, since what matters the most is how many combinations can occur with their use.

9. CONCLUSION

We analyzed the effectiveness of two identical AIs programmed to interact in English and Sanskrit in this research paper. After carefully studying them, we confidently conclude that Sanskrit, if used as a language to interact with the AI, performs brilliantly with minimum possible complexities.

With our study, we understand that Sanskrit, when used as a medium of communication with the AIs, proves to be much more fruitful than what we initially expected.

The Sanskrit language empowers the AI with more randomness, versatility and uncertainty as compared to the AIs programmed to interact in other languages. Besides this, it also equips the AI to follow a learning curve which is significantly steeper than that of AIs of any other language.

So Sanskrit could be used to make an evolved AI which is potent and powerful enough to simulate real world problems. This AI will be in no way sophisticated when compared with its opponents. This will not only make the AI more effective, but also keep the space and time complexities to optimum.

It is important to accept the fact that Sanskrit is not special when used as a programming language but is incredibly productive when used as a medium of interaction.

We have understood the role Sanskrit language is capable of playing in AI programming when applied as the medium of interaction with machines.

The source codes of all the AIs and programs used in this research paper have been uploaded on github repository under GNU license and are freely available for download from the web address given below-
https://github.com/Neetesh11/Evolvia-and-Arti.git

REFERENCES

[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville, (2016), "Deep Learning (Adaptive Computation and Machine Learning series)", *The MIT Press*, Massachusetts

[2] Michael Sipser, (2007), "Introduction to the Theory of Computation", Cengage Learning, Boston, United Sates.

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 17 – 27

[3] P. D. Turney, (2008), "The Latent Relation Mapping Engine", *Journal of Artificial Intelligence*, Vol. *33*
[4] Christopher Michael Bishop., (2007), "Pattern Recognition and Machine Learning", Springer, United States.
[5] Geoffrey E. Hinton, Simon Osindero, (2006), "A fast learning algorithm for deep belief nets", *ACM Digital Library.*
[6] Alan Mackworth, David Poole, (2010), "Artificial Intelligence: Foundations of Computational Agents", Cambridge University Press, United Kingdom.
[7] Henri Prade, Lluis Godo, (2012), "Weighted Logics for Artificial Intelligence", 20th European Conference on Artificial Intelligence Montpellier.
[8] Richard A. Frost, (2006), "Realization of Natural Language Interfaces using lazy functional programming", *ACM Computing Surveys.*
[9] T.J.M. Bench-Capon, Paul E. Dunne, (2007), "Argumentation in artificial intelligence", *Science Direct*, Vol. *171*
[10] T. Alsinet, C. Chesnevar, L. Godo, S. Sandri, (2006), "Modeling defeasible argumentation within a possibilistic logic framework with fuzzy unification", *IPMU*, Vol. 11, pp. 1228–1235
[11] Stephen Muggleton, Alireza Tamaddoni-Nezhad, Francesca A. Lisi, (2012), "Inductive Logic Programming", *21st International Conference*, Vol. 7270
[12] Kevin C. Desouza, (2002), "Managing Knowledge with Artificial Intelligence: An Introduction with Guidelines for Nonspecialists", *Praeger*, United States.
[13] Stuart Russell, Peter Norvig, (2009), "Artificial Intelligence: A Modern Approach", Prentice Hall; 3 edition, United States. cvc

Author's Biography

Neetesh Vashishtha is a UG student pursuing Computer Science Engineering in Jaipur Engineering College and Research Center, Jaipur, Rajasthan. He is currently working on simplification methodologies in AI programming with the application of conventional programming languages