Alternative Programming language for Indian Language Processing: A Choice among Java and C#

Harjit Singh[1], Ashish Oberoi[2]
[1]*Neighbourhood Campus Dehla Seehan, Punjabi University Patiala, Punjab, India*
[2]*School of Engineering, RIMT University, Mandi Gobindgarh, Punjab, India*
(e-mail: hjit@live.com)

Abstract**—**A natural language other than English needs to be processed using a programming language that is capable to handle Unicode character set. In natural language processing research, most of the researchers prefer to use Python language. But Java and C# can also be used to process a natural language because both these languages are capable to handle Unicode character set. These are suitable for processing Indian language text. These have built-in library functions that can be used to access and manipulate Unicode text strings. Due to the limited resources and limited ready to use libraries available for Indian language processing in other programming languages also, Java and C# are not so behind the race. Also the existing libraries and functions are capable enough to generate new libraries for Indian language processing. The features like object-oriented, structured programming, rich library and fast speed make both these languages suitable for Indian language text processing. This paper provides details about suitability of Java and C# for Indian language processing with the available Natural Language Processing toolkits and their support for Java and C#.NET including alternative options that will be helpful to beginners in Natural Language Processing.

Keywords**—** *Natural Language Processing, Indian Languages, Stanford CoreNLP, NLTK, NltkNet, OpenNLP, SharpNLP, Java, C#.NET*

I. INTRODUCTION

Indian language processing appears in beginning stage when compared to natural language processing (NLP) research for global languages. If we are just followers and want to follow the same pattern of research done for global languages then we should carry on otherwise at this initial stage it is not necessary to choose a programming language that others have chosen. Yes, the compulsion may be due to the reason that the research guide is unable to provide guidance in some other programming language and said that this particular language is best of NLP. I think the best language is the language we know the best and is capable to handle Unicode.

A natural language other than English needs to be processed using a programming language that is capable to handle Unicode character set. To implement Unicode, different character encodings are used. UTF-8, UTF-16, UTF-32 and many other encodings are used to represent Unicode character set. Unicode provided a consistent encoding and ways to handle text expressed in most of the natural languages written in the world. Unicode Consortium maintains the standard and the current version (as on March 2019) is Unicode12.0 which represents 1, 37, 929 characters [1]. The Unicode standard has been implemented in most of the recent technologies in the computing industry which includes operating systems like Windows, Meta languages like XML, and programming languages such as C #.NET.

Each Indian language has its own range of character representation in Unicode. For example, Devanagri Script (used to write Hindi language) has Unicode range from 0900-097F [2] and Gurmukhi Script (used to write Punjabi language) has Unicode range from 0A00-0A7F [3]. A number of toolkits available to process English language text but no up to the mark tool is available for Indian Languages text processing.

The basic functions in natural language processing are to convert Unicode character stream into set of lexical items such as words or phrases [4]. Both Java and C#.NET provide functions/methods to perform these tasks with some variations in working.

The remaining sections in this paper provide detailed analysis of functions and capability of Java and C#.NET and their suitability to process Indian language text. Some suggestions are given from personal viewpoint regarding the choice of a programming language while beginning the research in Indian language processing. The paper provides details about the available Natural Language Processing toolkits and their support for Java and C#.NET including alternative options. Finally the paper concludes with a summary of details presented in the paper and suggested opinions.

II. INDIAN LANGUAGE PROCESSING AND JAVA

Java is an object oriented language extended for software development and Internet applications. Java features such as platform independence, multithreaded application development, distributed application development, shows its suitability for development of up to date applications [5]. Its support for Unicode makes it useful for text processing of languages other than and including English [6]. This section

discusses various features of Java and their suitability for Indian Language Processing:

A. Platform Independence

If we require programming language code need to be platform independent so that it could be executed on any operating system or hardware, then Java is nice in this way. The model of java satisfies this requirement with the invention of Java Virtual Machine (JVM) or JRE (Java Runtime Environment). The design of java offers two phase compilation of java source to native machine code. When we compile Java source code, it outputs Bytecode which is an in-between code among java source code and the machine code. This Bytecode produced is interpreted to machine language code by the java interpreter. The standalone interpreter helps to execute the java application from the command line. This function is also performed by JRE (Java Runtime Environment). It means that to compile and execute java program, the machine must have Java Runtime Environment installed [7].

Java provided a new idea of improving performance using Just-In-Time (JIT) compilation (called HotSpot) that allows the Bytecode to be compiled to machine language code when it is needed and then store it in the memory so that if it is required again, it need not be re-compiled [8].

B. Speedup Processing

Indian Language Processing makes use of large set of data or text files (such as lexicons or corpus) which needs fast processing so as to produce outputs as quickly as possible. The Java architecture including JIT (Just-In-Time) compilation model makes it appropriate for these applications. Beyond that, Java feature called multithreading is very useful one to speed up processing of heavy burdened applications [8].

The design of an application for Indian Language Processing can be in such a way to split the tasks into multiple threads and those threads can be executed concurrently to lessen the processing time. For example, one thread can access the lexicon and another thread can interact with user requests and responses [9].

C. Easy Linguistic Knowledge Updation

Indian Language Processing requires a large set of linguistic knowledge collected in data files. Java is capable to access any database with same programming code. Java Database Connectivity (JDBC) establishes a bridged connection among Database and Application. Database of Linguistic knowledge remains independent from the programmed knowledge. The linguistic knowledge is updatable independently from the java code. The updated knowledge base is incorporated by java application without any change to answer future queries. Other applications are also allowed to update the linguistic knowledge base without any affect on Java application. The concurrent access to the linguistic knowledge stored in the database can be provided to various applications [10].

D. Linguistic Knowledge Reusability

The Linguistic knowledge utilized by Indian Language Processing application should not be restricted to access by a single application only. That linguistic knowledge need to be an independent set of data or information collected in such a way so that it can be accessed by any application.

In view of the fact that linguistic knowledge collected in the database is absolutely separated from related Java programs, it means that the knowledge is re-usable. The fresh applications can be developed to access the linguistic knowledge without any need to collect and store information from scratch. The Database stores the linguistic knowledge in tables, and more tables can be stored to existing database for new applications to be written without changing the working of existing applications [10].

E. Online Availability of Solutions

The utilization of Indian Language Processing systems can be increased by providing their access online instead of the need to have the full application installed on individual machine. Due to the heavy load applications, not all systems may fulfill installation requirements. Java's distributed application development is helpful to fulfill the need. Java Applets embedded in a web application page are executed in the web browser. Applets are capable to perform all tasks online similar a standalone application performs offline. An applet has an interface and according to the instructions by the user, it communicates with the web server to fulfill user requests in a safe way. That is the reason behind the extensive use of applets in online transaction applications. Java applet offers full utilization of java language strength in a web page [11].

III. INDIAN LANGUAGE PROCESSING AND C#

C# is also an object oriented language for software development and Internet applications (ASP.NET). C# features such as object oriented, type safety, interoperability, scalability, structured programming, fast speed and rich library shows its suitability for development of up to date applications [12]. Its support for Unicode makes it useful for text processing of languages other than and including English [13]. This section discusses various features of C# and its suitability for Indian Language Processing:

A. Platform Independence

A language is platform independent if the program compiled on one platform can be run without recompilation on some other platform. The architecture of C# is similar to Java in some way. The Java programs when compiled require JRE (Java Runtime Environment) in order to execute in the same way C# programs when compiled require CLR (Common Language Runtime) in order to execute. The design of C# (or we can say .NET) offers two phase compilation of source code to native machine code. When we compile C# source code, it outputs CIL (Common Intermediate Language) code which is an intermediate code between C# source code and the machine code. This CIL produced is interpreted to machine language code by the CLR (acting as interpreter). It means that to

compile and execute C# program, the machine must have .NET Framework (CLR is a part of) installed [14].

C# also uses the idea of improving performance using Just-In-Time (JIT) compilation but with some differences. Here the CLR compiles the whole code when it is invoked at runtime. It improves performance in case large amount of code is there and calls to same code are less frequent [15].

B. Speedup Processing

Indian Language Processing makes use of large set of data or text files (such as lexicons or corpus) which needs fast processing so as to produce outputs as quickly as possible. The .NET architecture (hence C#) including JIT (Just-In-Time) compilation model makes it appropriate for these applications. Beyond that, C# also provides multithreading which is very useful one to speed up processing of heavy burdened applications [15].

The design of an application for Indian Language Processing can be in such a way to split the tasks into multiple threads and those threads can be executed concurrently to lessen the processing time. For example, in case of translation and transliteration one thread can deal with source language and another thread can deal with target language [16].

C. Easy Linguistic Knowledge Updation

Indian Language Processing requires a large set of linguistic knowledge collected in data files. C# is capable to access any database with same programming code. ADO.NET establishes a connection among Database and Application using Entity Framework. Entity Framework is an open source object-relational mapping framework for ADO.NET. Using Entity Framework, we can work at a higher level of abstraction while dealing with data. Database of Linguistic knowledge remains independent from the programmed knowledge. The linguistic knowledge is updatable independently from the source code. The updated knowledge base is incorporated by C# application without any change to answer future queries. Other applications are also allowed to update the linguistic knowledge base without any affect on C# application. The concurrent access to the linguistic knowledge stored in the database can be provided to various applications [17].

D. Linguistic Knowledge Reusability

The Linguistic knowledge utilized by Indian Language Processing application should not be restricted to access by a single application only. That linguistic knowledge need to be an independent set of data or information collected in such a way so that it can be accessed by any application.

In view of the fact that linguistic knowledge collected in the database is absolutely separated from related C# programs, it means that the knowledge is re-usable. The fresh applications can be developed to access the linguistic knowledge without any need to collect and store information from scratch. The Database stores the linguistic knowledge in tables, and more tables can be stored to existing database for new applications to be written without changing the working of existing applications [17].

E. Online Availability of Solutions

The utilization of Indian Language Processing systems can be increased by providing their access online instead of the need to have the full application installed on individual machine. Due to the heavy load applications, not all systems may fulfill installation requirements. .NET Remoting provides distributed application development which is helpful to fulfill the need. ASP.NET web applications also provide communication with web server through web browser. A web page has an interface and according to the instructions by the user, it communicates with the web server to fulfill user requests in a safe way [18].

IV. THE READY TO USE TOOLKITS

There are a number of Natural Language Toolkits available to use with various programming languages. But the support for Indian Languages is very limited till now:-

A. Standford CoreNLP

Standford CoreNLP is a set of NLP tools which take raw text in English and outputs base word forms, the POS (Parts-of-Speech) tags of words, recognizes named entities etc. It means it is easy to apply these tools to the plane text with a few lines of code. It is very flexible and extensible. In the tool pipeline, we can choose which tools we want to apply (enabled) and which tools we do not want to apply (disabled).

Java language is used to write Stanford CoreNLP, so Java needs to be installed to run it. Although it is better to use it from a Java application, but the access is not limited to Java. There are APIs available to access it from other programming languages such as C#.NET, F#.NET, Go, Perl, PHP, Python, Ruby and many others. Even it can be accessed as a web service or from command line [19].

CoreNLP is basically expected for processing English language text. But its architecture allows it to work with other natural languages, given that there are components/models available for those languages. The first party language support is provided to English, French, German, Spanish, Chinese, Arabic and unfortunately not to any Indian language till now [20]. So, using CoreNLP for Indian languages is as difficult as starting from scratch.

B. NLTK

Natural Language Toolkit (NLTK) is a popular package of tools to work with natural languages using Python. It provides libraries for text processing such as tokenization, classification, tagging, stemming, parsing etc. and interfaces t more than 50 corpora including interface to WordNet. It can be used on Windows, Linux, Mac OS X and it is open source and free. It is useful for researchers, linguistics, students and educators [21].

Stemming is a basic task in Natural Language Processing. NLTK 2.0.4 stemming supports only these languages i.e. English, French, Italian, German, Spanish, Russian, Swedish, Porter, Portuguese, Hungarian, Norwegian, Romanian, Dutch, Danish, Arabic and Finnish. It should be cleared that not all functionalities are available for all supported languages. So, there is no Indian languages support from first party. Also there

are no POS taggers that are pre-trained for languages other than English. NLTK is a great tool for teaching and research in supported languages but it is quite limited if used for industrial applications [21].

C. NltkNet

It is a .NET wrapper for NLTK library and hence enables support of NLTK for c# language. But to use NltkNet wrapper, we need to install IronPython binaries and then NLTK library for IronPython. IronPython interpreter is very useful to execute and test Python scripts from Visual Studio.NET. After successful installation and confirmation, NLTK can now be accessed from Visual Studio.NET using C# as follows [22]:

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using NltkNet;

namespace NltkNetTest
{
 class TestClass
 {
  static void Main(string[] args)
  {
        Nltk.Init(new List<string>
        {
            @"X:\path-to-IronPython-libraries",
            @"X:\path-to-third-party-libraries",
       });
         string text = "Testing NltkNet using C# on VS.NET.";
       var words = Nltk.Tokenize.WordTokenize(text);
      // ----Statements----
      var corpus = new Nltk.Corpus.Brown();
     // ----Statements----
     var stemmer = new Nltk.Stem.PorterStemmer();
     // ----Statements----
  }
 }
}
```

But unfortunately, NltkNet is new and does not support all features of NLTK library. For those unwrapped features, we can use Nltk.Py directly from C# code to execute IronPython script as [22]:

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using NltkNet;

namespace NltkNetTest
{
 class TestClass
{
 static void Main(string[] args)
 {
        Nltk.Init(new List<string>
        {
            @"X:\path-to-IronPython-libraries",
            @"X:\path-to-third-party-libraries",
         });
        NltkNet.Py.ImportModule("nltk.corpus");
        NltkNet.Py.ExecuteScript("from nltk.corpus xxxx");
 }
 }
}
```

D. OpenNLP

OpenNLP is a toolkit written in Java to process natural language text based on machine learning. Common text processing tasks like sentence segmentation, tokenization, POS tagging, named entity recognition (NER), parsing, chunking etc. are supported by OpenNLP. It supports perception based machine learning and maximum entropy based machine learning framework for training the NLP components, so it can support a natural language given that the language model is available. The latest release of OpenNLP (as on March 27, 2019) having Language Detector Model is able to detect 103 natural languages which includes 9 Indian languages i.e. Hindi, Punjabi, Gujarati, Marathi, Malayalam, Sanskrit, Tamil, Telugu and Urdu [23].

E. Sharp NLP

SharpNLP is a set of open source natural language processing tools developed in C# for English language processing. It is related to OpenNLP library and provides natural language tools such as sentence splitter, POS tagger, tokenizer, chunker, parser, named entity finder etc. it also provides an interface to WordNet database [24].

V. CONCLUSION

There are a number of options available for a researcher who begins research in Indian languages processing. No programming language can be stamped as best for processing language text. Actually the language suggestion is just related to the availability of ready to use libraries for speeding up the research process. As discussed in this paper, it is clear that Indian languages are far away from language support by first party NLP tool developers. So, taking into this fact in consideration, one must use his/her expertise in that particular language which he/she knows the best. C#.NET is a modern programming language and can be used in Indian language processing as well since no mature libraries exist for Indian language processing in other programming languages also. The beginning in an experienced programming language will be

better for concentrating on the research ideas instead of learning a new language and putting the research work aside. This paper provided details about the suitability of C# for Indian language processing alongside Java. The paper also provided lights on available NLP toolkits and their support for Java and C#.NET including alternative options, so that beginners in research can understand the things better at one place.

REFERENCES

[1] UNICODE INC. Announcing The Unicode® Standard, Version 12.0. [Online]. http://blog.unicode.org/2019/03/announcing-unicode-standard-version-120.html

[2] UNOCODE INC. Unicode range for Devanagri. [Online]. http://www.unicode.org/charts/PDF/U0900.pdf

[3] UNICODE INC. Unicode range for Gurmukhi. [Online]. http://www.unicode.org/charts/PDF/U0A00.pdf

[4] Eiichiro Sumita Takao Doi, "Input Sentence Splitting and Translating," in Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond, 2003, pp. 104–110.

[5] Harjit Singh, "Role of Java in Natural Language Processing for Indian Regional Languages," International Journal of scientific research and management (IJSRM), vol. 4, no. 11, pp. 4800-4802, Nov. 2016.

[6] ORACLE INC. Unicode (The Java™ Tutorials). [Online]. https://docs.oracle.com/javase/tutorial/i18n/text/unicode.html

[7] Marisa Gil Ruben Pinilla, "JVM: Platform Independent vs. Performance Dependent," ACM SIGOPS Operating Systems Review, vol. 37, no. 2, pp. 44-56, April 2003.

[8] IBM. JIT Overview (Java). [Online]. https://www.ibm.com/support/knowledgecenter/en/SSYKE2_8.0.0/com.ibm.java.vm.80.doc/docs/jit_overview.html

[9] ORACLE INC. Processes and Threads (The Java™ Tutorials). [Online]. https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html

[10] ORACLE INC. JDBC Introduction (The Java™ Tutorials). [Online]. https://docs.oracle.com/javase/tutorial/jdbc/overview/index.html

[11] ORACLE INC. An Overview of RMI Applications (The Java™ Tutorials). [Online]. https://docs.oracle.com/javase/tutorial/rmi/overview.html

[12] Microsoft Corp. Introduction to the C# Language and the.NET Framework. [Online]. https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

[13] Microsoft Corp. Character Encoding in.NET. [Online]. https://docs.microsoft.com/en-us/dotnet/standard/base-types/character-encoding

[14] CODE PROJECT. Understanding Common Intermediate Language-CIL. [Online]. https://www.codeproject.com/Articles/362076/Understanding-Common-Intermediate-Language-CIL

[15] Microsoft Corp. Managed Execution Process. [Online]. https://docs.microsoft.com/en-us/dotnet/standard/managed-execution-process

[16] Microsoft Corp. Managed Threading. [Online]. https://docs.microsoft.com/en-us/dotnet/standard/threading/

[17] Microsoft Corp. ADO.NET Overview. [Online]. https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview

[18] C-Sharp Corner. Remoting in.NET. [Online]. https://www.c-sharpcorner.com/article/remoting-in-net/

[19] Stanford University. Stanford CoreNLP – Natural language software. [Online]. https://stanfordnlp.github.io/CoreNLP/

[20] Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky Manning, "The Stanford CoreNLP Natural Language Processing Toolkit," in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2014.

[21] NLTK Project. Natural Language Toolkit. [Online]. https://www.nltk.org/

[22] NltkNet. [Online]. https://github.com/nrcpp/NltkNet

[23] Apache. OpenNLP. [Online]. https://opennlp.apache.org/

[24] CodePlex. SharpNLP. [Online]. https://archive.codeplex.com/?p=sharpnlp

Authors Profile

Mr. Harjit Singh received the MCA (Master in Computer Applications) degree from IGNOU (Indira Gandhi National Open University), New Delhi, India and M.Phil.(CS) degree from Global Open University, Nagaland, India. He is working as Assistant Professor (Senior Scale) in Computer Science at Punjabi University Neighbourhood Campus Dehla Seehan, Sangrur, India. He is pursuing Ph.D. degree from RIMT University, Mandi Gobindgarh (Punjab). His current research interests include Natural Language Processing, Machine Translation, Artificial Intelligence.

Dr. Ashish Obroi is working as Professor in Computer Science and Engineering at School of Engineering, RIMT University, Mandi Gobindgarh, Punjab, India. He completed his Ph.D. in Computer Science and Engineering from Maharishi Markandeshwar University, Mullana, India. He is skilled and expertise in Image Processing, Medical Imaging, Image Segmentation, Diagnostic Imaging and Image Reconstruction.