International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 09 | Sept - 2021 ISSN: 2582-3930

SURVEY PAPER ON A SMART CHATBOT MODULE

BHAVANA H V

M tech IV SEM(CS &E)

Department of computer science and engineering

The university B D T College of engineering, Dvanagere -577004,Karnataka ,India

(A constituent college of Vivesvaraya Technological University ,Belagavi)

DR. MOHAMED RAFI

Department of computer science and engineering

The university B D T College of engineering, Dvanagere -577004,Karnataka ,India

(A constituent college of Vivesvaraya Technological University ,Belagavi)

ABSTRACT: This paper presents the design and development of an intelligent voice recognition chat bot. The paper presents a technology demonstrator to verify a proposed framework required to support such a bot (a web service). While a black box approach is used, by controlling the communication structure, to and from the web-service, the web-service allows all types of clients to communicate to the server from any platform. The service provided is accessible through a generated interface which allows for seamless XML processing; whereby the extensibility improves the lifespan of such a service. By introducing an artificial brain, the web-based bot generates customized user responses, aligned to the desired character. Questions asked to the bot, which is not understood is further processed using a third-party expert system (an online intelligent research assistant), and the response is archived, improving the artificial brain capabilities for future generation of responses.

Index Terms : AI, XML, JAVA, AIML, ALICE.

I. INTRODUCTION: A chatbot is a software program of a conversational interface that allows a user to converse in the same manner one would address a human. A virtual chatbot is a piece of software that is intelligent enough to mimic human interactions. Conversational bots are used in almost every customer interaction, like instantly messaging the client. Since the development of the first chatbot, they have evolved

in functionality, interface, and their significance to the technical world cannot be neglected. However, modelling conversations remains a significant challenge in this field even today. Even though they are a long way from perfect, conversational agents are now used in various applications [1]. To understand the capabilities and limitations of current chatbot techniques and architectures, a detailed survey was conducted, where interrelated literature published over the past few years is studied, and a newly presented neural network model is now trained with conversational data. Deep learning and NLP techniques are being used in advanced research and development projects, AI and ML algorithms are implemented in development of conversations. R&D (Research and Development) are still under progress and experimentation in these fields. Conversation agents are predominately used by government administrations, businesses, and non-profit establishments. They are often organized by financial institutions like banking, insurance, start-up companies, online stores and social service sectors. These chatbots are implemented by large corporations as well as small start-up companies. However, chatbots are not under proper implementation in the medical field. A chatbot can help patients with medical-related works by assisting them via text

messages, applications, or instant messaging. One can find many virtual bot development structures in the market, both interface-based and code-based. However, both models have limitations concerning flexibility and practicality in making real conversations. Most of the well-known intelligent personal assistants like Alexa by Amazon, Google Assistant by Google, and Cortana by Microsoft do have some limitations in functionality. Retrieval based agents are being introduced to hold conversions that match real interactions alike humans. Among many intelligent personal assistants currently available, they are structured on rule-based techniques or retrieval-based techniques that generate decent results [2]. Recently there is a significant increase in curiosity in the usage of chatbots. Many companies are successfully using these dialogue generation devices to full fill customer requirements. Since companies are adopting the chatbot technology, there is a promising demand for the development and advanced research for the conversational agents.

II . RELATED WORK Chatbot aims to make communication between a human and machine such as computer and mobile. Recently a considerable amount of promising work has been conducted in the area of chatbot design. O. V. Derouging presented a detailed survey on the history of the chatbot, their applications, and the first designs of such systems. Bores et. al , presented an intelligent question answering system which achieved competitive results. They trained their model using low-dimensional embedding of words and knowledge base constituents and used these representations to score natural language questions against candidate answers. Pareira et.al presented an overview of the chatbot early contributions and tried to map those with the current works in the humanmachine interaction research. We have noticed that the chatbot

related research is mainly distributed in the following areas, (I) different approaches (e.g., retrieval and generative), (ii) length of the conversation, and (iii) according to the domain (e.g., open and closed). Retrieval-based models use a repository of predefined responses and a heuristic to pick an approve on the input and context . The heuristic could be as simple as a rule-based expression match or as complex as an ensemble of machine learning classifiers. The process of machine learning is similar to that of data mining. Both systems search through data to look for patterns. Besides, generative models don't rely on predefined responses. They generate new responses from the scratch. Generative models are typically based on Machine Translation techniques where we" translate" from an input to an output (response). Both approaches have some obvious pros and cons. Due to the repository of handcrafted responses, retrieval-based methods don't make grammatical mistakes. However, they may be unable to handle unseen cases for which no appropriate predefined response exists. For the same reasons, these models can't refer back to contextual entity information like names mentioned earlier in the conversation. Generative models are" smarter", however, these models are hard to train, are quite likely to make grammatical mistakes (especially on longer sentences), and typically require huge amounts of training data. The longer the conversation, the more difficult to automate it. On one side of the spectrum are Short Text Conversations (easier) where the goal is to create a single response to a single input. In an open domain setting, the user could take the conversation anywhere. There isn't necessarily have a well-defined goal or intention. The infinite number of topics and the fact that a certain amount of world knowledge is required to create reasonable responses makes this hard problem. On the other-side, a closed domain setting, the space of possible inputs and outputs is somewhat limited because the system is trying to achieve a very specific goal.Technical customer support or shopping assistants are examples of closed do`main problems.

- **Natural Language Processing Models** : If Else Chatbots
  - If-else Chatbots are one of the earliest techniques which was used to create Chatbots. In these Chatbots, a huge List of questions and answers-based script was required.
  - The Chatbots returned an assigned answer to the question if the entire question or at least majority of the String matches the already saved question. • The problems were

1. The answers were repetitive, and the reply gave a mechanical feel.

2. A huge script was needed to make the chatbot, covering a variety of questions, and even after that, the chatbot made was very organization dependent.

3. The chatbot could not cope with growing data, as the script size grew the system started taking more time to generate an answer.

❖ Bag of Words Model

• Bag of words is a great model for extracting features from a text or use in modelling.

• Bag of words is a representation of text counting the occurrence of the key words/words.

• A bag-of-words represent the text by involving two things. A vocabulary of known words. A measure of the presence of known words.

• The limitations of the bag of words model are as follows **Sparsity** how the harness so raw information can be represented in a large space.

Vocabulary: it affects the sparsity of the document/word. **Meaning**: semantic of the words should help a lot. Context and meaning should be appropriate.

❖ Deep Natural Language Processing Models

Convolutional Neural Networks for text Recognition. Convolutional neural networks are neural networks which are mostly used in image recognition For NLP, the following steps are required:1. extract higher-level features using n-grams and constituting words 2. Tokenization and modelling of sentence of d dimension. 3. Applying Convolutional filters on inputs called a feature map. 4. Max operation on each filter is applied; max-pooling operation to obtain a fixed length output and reduce the dimensionality of the output. The problem with the above approach is their inability to model long-distance dependencies.

• RNN is a special type of0 neural network which memorize the previous iteration result and store it in buffer to compute the result of next iteration. A RNN recursively do the computation to process the inputs and gives the desirable output.

• Many applications which take previously processed data and gives output based on past experiences and so forth. In these cases, RNN is the best approach to use and create a working model for chatbots.

Long Short-Term Memory (LSTM)

• An LSTMs can in principle use its memory cells to remember long-range information and keep track of various attributes of text it is currently processing.

• Three types An LSTM consists of three gates (input, forget, and output gates), and calculate the hidden state through a combination of the three.

• GRU are similar to LSTMs but consist of only two gates and are more efficient because they are less complex

• The consistent finding is that depth of at least two is beneficial. However, between two and three layers, our results are mixed. Additionally, the results are mixed between the LSTM and the GRU, but both significantly outperform the RNN.

• Hence Our model for the chatbot uses LSTM

III. SYSTEM ARCHITECTURE: The system consists of the following three components: client, server, and content acquisition. The server is a simple object access protocol (SOAP) aware internet application (web service) based on a black box approach. A black box approach isolates the client from interacting with the inner workings of the web service; as opposed to a white box approach, where the inner workings are essential

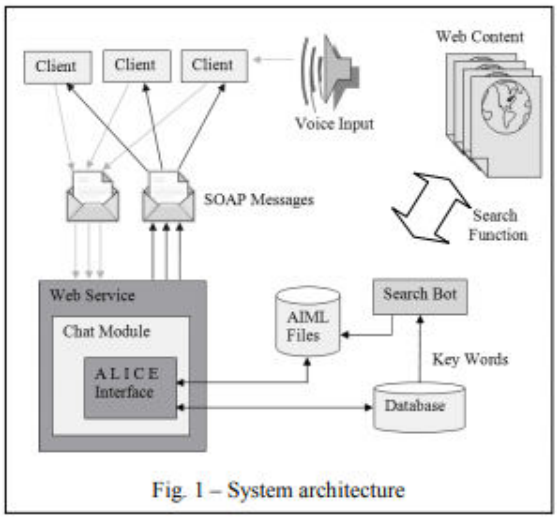

Fig. 1 – System architecture

and allows the client to interact with a distributed environment. As shown in Fig. 1, all messages are formatted in an extensible mark-up language (XML) and encapsulated as a SOAP message pack. The packs are text based allowing for a greater diversity of clients and platforms. The client contains the voice recognition processing module which allows the client to only send and receive plain text.

The web service processes all received queries using the response generation module (based on the Artificial Linguistic Internet Computing Entity (ALICE) [10] system), which makes use of a data repository. The data repository is updated by the content retrieval module to increase the intelligence autonomously (based on an

Artificial Intelligence Mark-up Language (AIML)). With this approach, an external administrator is only required to verify the quality of the self-training function. For a future query, the content is re-processed and incremental updates are made.

IV. SYSTEM SPECIFICATIONS: The system presented in this paper meets the following requirements. The client application is easily accessible through the client browser. All communications to and from the server is text and XML formatted. XML messages conform to a schema which describes the format. Communications with the server is black box oriented and parses incoming XML messages seamlessly. The user is allowed to register and login to the system allowing for authenticated, personalized and controlled communication with the server. The client applet provides the user two options: text input or voice input. • The self-training AI module prevents a service bottleneck, and therefore prevents modules from competing for resources.

Open source approach :There are a number of available libraries and open-source technologies to implement the specifications presented in this paper. This approach allows new implementations of a paradigm of using existing. The pre-loading of libraries allows for streamlined operation (i.e., the libraries are not called in an ad hoc basis, halting the sequence of activities). The user is prompted to accept a signature, upon such an acceptance; the applet can securely communicate to the web service. By using the open-source development environment NetBeans, the applet and the libraries can be digitally signed. This allows the applet to communicate with a web service not located on its source web server. The chat client is interrupt driven and activates upon interaction from a user. This is shown in Fig. 3; where the control unit decides what component to launch next and what function to process.

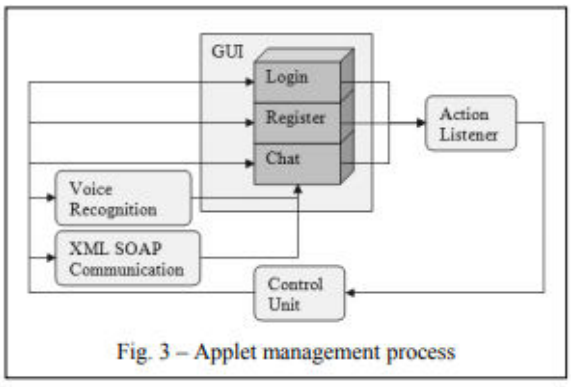

Fig. 3 – Applet management process

The action listener is bound to the buttons (login, register and chat) which is included in the graphical user interface (GUI). Once an event is triggered, either the corresponding SOAP communication module or voice recognition module or interface is activated. The input and output to these modules rely on user input captured

via the GUI. Voice recognition is accomplished using an open-source library called Sphinx 4. Speech recognition can be broken into two groups with five subsets in total; speakers and speech styles. Speakers make up single speaking and speaking independently where speech styles include isolated word recognition, connected word recognition and continuous word recognition. Isolated word recognition requires long breaks between all words to successfully interpret words. Continuous word recognition also requires this but considerably shorter breaks. The last subset of speech styles is continuous speech recognition where one can speak fluently and not require stopping or breaking between words. This has problems interpreting similar vowel in the beginning of words. A vital part of the speech system is contained within a configuration file. As shown in Fig. 4, the configuration file is used to build the recognizer and (or) decoder using the application programming interface (API), including the dictionary. Grammar files which is used in generating the message is an integral part forming part of the acoustic model located in the API

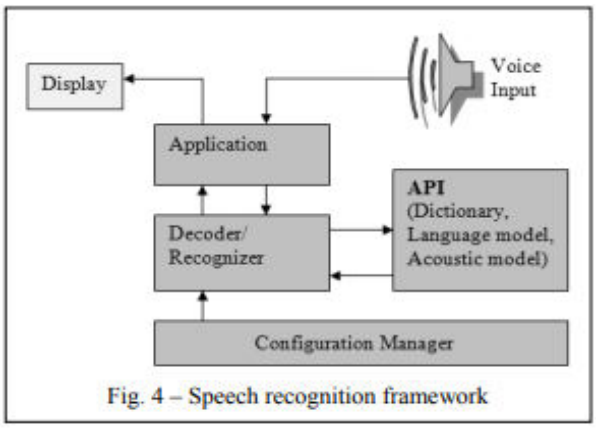

Fig. 4 – Speech recognition framework

The main components of the speech framework consist of the front-end application, decoder, and language model. The front-end acquires and attributes the voice input. The language and acoustic model is used to translate from a standard language (as input to the system) using a dictionary and construction of words located in a look-up table (LUT). A search manager located in the decoder then uses the attributes and LUT to decode the input voice into a result set. From the front-end, the user can activate the configuration manager, loading all components stored in XML format. When running a speech enabled application, the application requires more than 256 Mb in heap size thus placing excessive memory demands on the system Although a more toned down API (Pocketsphinx [9]), focusing on mobile platform is available, this paper focuses of Spinx 4. Communication with the web service is text based and XML-formatted using SOAP. The interface was generated using a JAVA-based architecture for XML binding (JAXB). An XML schema based API can also be published on a website for other developers to develop their own clients. This process of generating the interface is shown in

Fig. 5.

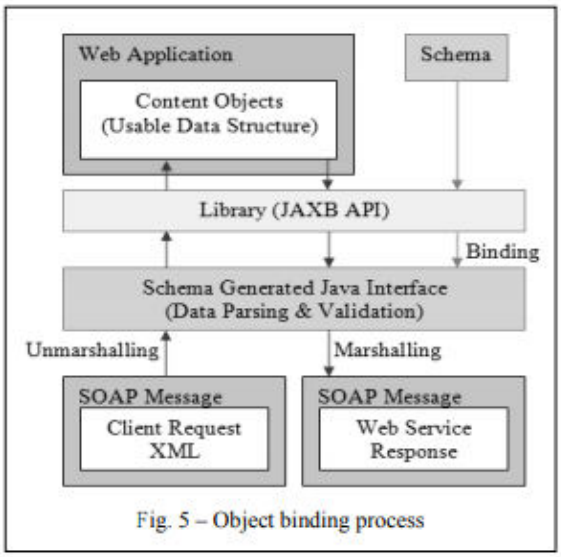

Fig. 5 – Object binding process

All messages are parsed through this interface which creates structure and validates the XML at the same time. The process of parsing the XML to a usable structure is termed marshalling. When an object is created, the XML content is handled using the object extraction through the interface class created on startup as indicated in the next code segment.

```
JAXBContext.newInstance jaxbContext = JAXBContext.newInstance( "icbxml" );
Unmarshaller u = jaxbContext.createUnmarshaller();
u.setEventHandler(new ICBXMLValidationEventHandler());
outputPipe.connect(inputPipe);
byte[] bytes = xmlData.getBytes();
outputPipe.write(bytes);
JAXBElement mElement = (JAXBElement)u.unmarshal(inputPipe);
IcbXml icbxmlvar = (IcbXml) u.unmarshal(inputPipe);
IcbXml po = (IcbXml)mElement.getValue();
```

The interface also provides fault response generation if a XML message is invalid. The error location is then encapsulated in XML using the bound object and sent using SOAP. The interface can be regenerated on demand if the requirements or format of the message format changes. The server maintains a large number of users through the process of synchronized threads. The most basic hypertext transfer protocol (HTTP) connection or request for content is done through sockets. When a client connects through a basic connection request (socket connection) the socket creates a new thread pair (incoming, outgoing and processes) to handle the client messaged. All messages pulled from the socket are then processed through the generated interface object. The thread pair queues all messages though a synchronized push and pop principle. Messages are pulled from the queue for processing in a controlled fashion, as shown in Fig. 6. This optimizes processing time. At the end of a queue, only the relevant thread would go to sleep for a pre-determined time period. This still allows the other threads to continue parsing messages in the queue and transmitting them out. When a user logs on, the system will by default greet the user, and then prepare to receive a question or a statement. ALICE [10] is an open source foundation developing AI chat systems. An implementation of the ALICE-bot engine, Chatterbean uses its algorithm to generate a response using its library for pattern assimilation (targeting), namely AIML files. The bot is centered on supervised human assistance for a learning process. This helps to remove or indicate correct answers as the system learns from conversations with users and updates the AIML files. AIML files are structured to consist of XML formatting, with context descriptions as indicated in the next code segment. How old are you Me I am as old as the mountains. Although a large set of AIML files are there to guide the bot it does not dictate a response. These sets of categories make up groups of questions and answers. Thus the intelligence is limited. If a question appears where no response can be generated (which will happen often in the infancy of an AI system) the system will try to change the topic or send a general statement. This is considered a trick for inadequate knowledge but is a first step for any chat bot. To assist in the training, a training module was created to process content acquired from the internet related to statements or questions which the system cannot understand. This would require a complex algorithm to not not only search for related content source but provide for concise content. Thus a third party

expert system, "Ultimate Research Assistant" [11] was used to generate a detailed report relating to such a statement. This process is shown in Fig. 7. To get the most relevant response from the expert system, the query sent to it needs some refinement. To refine the query, self processing using AIML-assisted categories is used. The process is methodic (Fig. 8), for instance, no result may be generated from the entered query "Who is Albert Einstein?," the processing proceeds iteratively to secondary file(s), where the same query is posed. The result of this query may be "a human icon," the processing proceeds, and the resultant response is a combination of the output seeked from the secondary file and a standard response ("I am sorry… provide the right questions.")

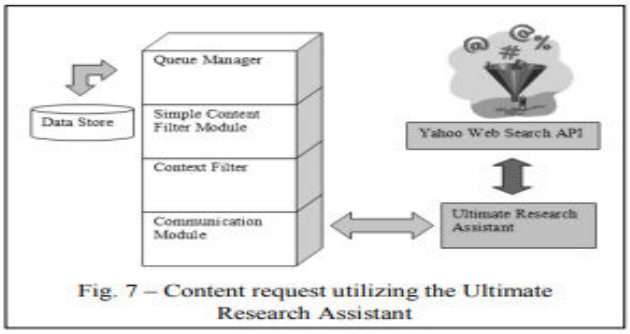

Fig. 7 – Content request utilizing the Ultimate Research Assistant

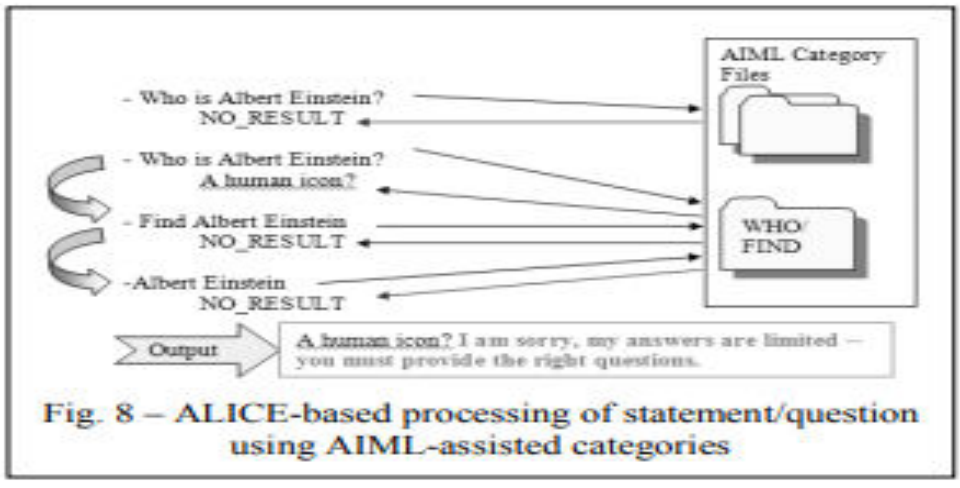

Fig. 8 – ALICE-based processing of statement/question using AIML-assisted categories

In the particular instance, “Albert Einstein” is not understood, but also becomes the refined query. The refined query is then queued as an input to the expert system. As the queue is processed, reports are generated, which further populates the AIML files. This process is shown in Fig. 9.

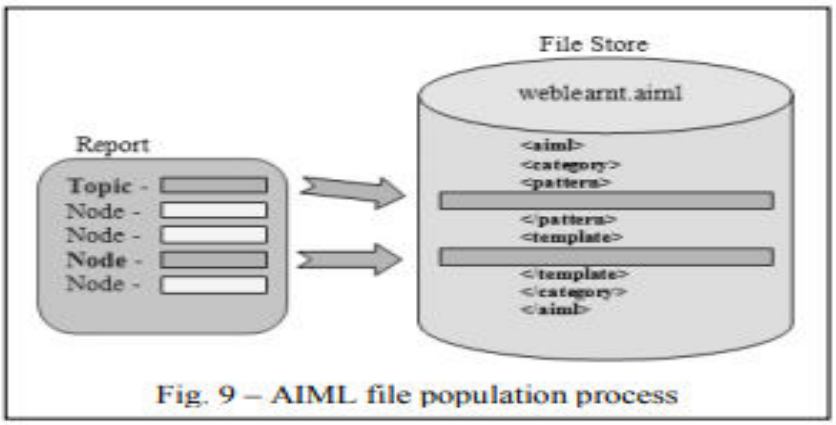

Fig. 9 – AIML file population process

The process of Fig. 9 is autonomous and needs to be moderated minimally to control the AIML file expansion. This process results in improving the intelligence of the system.

RESULTS

The combination of voice input and voice output allows for a simpler experience which allows a client to run on many types of platforms. Since the client is internet based the next step would be mobile or even thin client systems. A thin client system is considered an embedded computer or platform with limited processing where the processing is done by a controlling server. Examples include a mini computer on a fridge, GPS unit or a mobile phone. Although the aim of this paper was to implement an intelligent virtual online friend, with voice, this integration of technologies can also be used in other applications, particularly when dealing with a need for simple input control accessibility and/ or limited processing capabilities. The system resulted in a distributed environment to allow for resource management and stability between modules. This is shown in Fig. 10, handling the hosting of the site, processing responses using the ALICE-bot engine, content acquisition and processing using an expert system to increase the intelligence of the chat bot autonomously. The use of the distributed framework allows for an increase in throughput and the number of users it can handle. The lifetime of the expert system can be a limitation to the age of the technology demonstrator presented in this paper.

VII. CONCLUSION Using modular design for all its components a distributed environment facilitating transparent and high performance of the overall system has been created. The performance is relative to the

processing capacity of the systems involved. Since all the modules are not running off one system the possible load has been decreased and further decreased by delegating the voice processing to the chat client communicating with the service.

REFRENCES

[1] Salomon Jakobus du Preez completed his National Diploma in Technical Applications from the Tshwane University of Technology (TUT), South Africa in 2007. Manoj Lall completed his B.Eng (Mechanical) and M.Sc. (2005) in Computer Science from the University of South Africa, Saurabh Sinha (M'02) completed his B.Eng degree (cum laude), M.Eng degree (cum laude) and PhD(Eng) degree from the University of Pretoria, South Africa, in 2001, 2005, 2008 respectively, AN INTELLIGENT WEB-BASED VOICE CHAT BOT S. J. du Preez1 , Student Member, IEEE, M. Lall2 , S. Sinha3 , MIEEE, MSAIEE

[2] A Deep Neural Network Based Human to Machine Conversation Model G Krishna Vamsi Computer science and engineering Maulana Azad National Institute Of Technology Bhopal, India Akhtar Rasool Computer science and engineering Maulana Azad National Institute Of Technology Bhopal, India Gaurav Hajela Computer science and engineering Maulana Azad National Institute Of Technology Bhopal, India

[3] Chatbot: An automated conversation system for the educational domain **Anupam Mondal** Computer Science and Engineering Jadavpur University Kolkata, India link.anupam@gmail.com **Monalisa Dey** Computer Science and Engineering Jadavpur University Kolkata, India monalisa.dey.21@gmail.com Dipankar Das Computer Science and Engineering Jadavpur University Kolkata, India dipankar.dipnil2005@gmail.com 4 th Sachit Nagpal Big Data and Analytics S P Jain School of Global Management Mumbai, India sachitnagpal@gmail.com 5 th Kevin Garda Big Data and Analytics S P Jain School of Global Management Mumbai, India kevingarda7@gmail.co