International Journal For Technological Research In Engineering
Volume 6, Issue 7, March-2019 *ISSN (Online): 2347 - 4718*

INTROSPECTION OF NATURAL LANGUAGE PROCESSING FOR AI CHATBOT

Snehal Kulthe[1], Mrs. Vineeta Tiwari[2], Mr. Manish Nirmal[3], Prof. Bhushan Chaudhari[4]
[1,4]Department of Computer Science, Sandip University, Nashik, India
[2,3]Department of ACTS, CDAC ACTS, Pune, India

Abstract: ***With Alan Turing's question "Can Machines Think?" chatbot concept came into existence. Since Turing, the ability of machines has been challenged. In the era of Artificial Intelligence advances in Natural Language Processing, Machine Learning led to the adoption of the Chatbots. Machines have started to mimic human traits. Artificial Intelligence conversational entities, also called chatbots are computer programs capable to carry out near-natural conversation with people. For a chatbot to perfectly emulate a human dialogue, it must examine the input given by a user correctly and articulate a relevant response. This paper presents a literature survey of the chatbot architecture, design, and algorithms used in chatbots. In this paper, we also describe the evolution of chatbots from a primitive model of a smart system. The aim of this survey paper is to build a Chatbot for interview.***

Keywords: Chatbots, Machine learning, Natural Language processing, Deep Learning, RNN, NLU, NLG, NLP

I. INTRODUCTION

Chatbots of 2018 are unamended since the first chatbot ELIZA in 1966. While recent advances in machine learning have contributed to faster improvements in conversational interfaces, chatbots are still not perceived as intelligent conversational actors. Since ELIZA, the goal of chatbot systems have been to pass the Turing Test, and convince humans that they are conversing with a human, not a machine.[1]

A progressing development in intelligent Chatbots has been observed since ELIZA. Kenneth Colby in 1972 at Stanford created a chatbot that impersonated a paranoid schizophrenic. In 1995, Richard Wallace created A.L.I.C.E, a complex chatbot that generated responses by pattern matching inputs. Artificial Intelligence Markup Language (AIML) was used to write this documents. Amazon's Echo and Alexa, Apple's Siri, and Microsoft's Cortana are the modern chatbots. The architectures and retrieval processes of these bots are such that the responses are generated based on analysis of the results of web searches. The "generative" models are used to respond; Statistical machine translation (SMT) is used techniques to "translate" input phrases into output responses. The sequence to sequence model uses recurrent neural networks (RNNs) to encode and decode inputs into responses. Through this research project, we first explore different architectures of ANNs for NLP and analyze the existing models used to build a chatbot. We then attempt to gain insight into the questions: How is a chatbot built as of today, what are their limitations and where can we try to improve? We also present a study of different linguistic models used so far in Natural Language Processing.

II. EVOLUTION OF CHATBOTS

A. ELIZA

The term 'Chatbot' was coined by Mauldin to define the systems aiming to pass the Turing Test. ELIZA in 1966 created by Weizenbaum was widely recognized as the first program capable to pass the Turing Test. ELIZA's model was specifically rephrasing input and trying to match keywords with pre-defined responses. When none of the rules matched ELIZA had some responses which were regardless of any context, in addition to rules. The main aspect that made ELIZA popular was because people thought they were talking to real human. ELIZA being a rule-based system, its conversations with humans were not much of meaningful. Also, as explained above, it had generic responses to articulation Where none of the rules were applicable.[3] Various chatbots were inspired by this simple rule-based framework, like PARRY and A.L.I.C.E.[3]

B. Jabberwacky

Rollo Carpenter, a British programmer designed Jabberwacky. It is meant to be a conversational bot making interesting and humorous conversation. It was one of the first chatbots which harnessed AI to learn meaningful human conversations. Jabberwacky won the Loebner prize, an annual competition to test chatbots, in 2005 and 2006. Jabberwacky was also able to learn continually from a conversation. It can learn new languages, concepts and facts just by constant interactions. The creators have maintained that Jabberwacky is not based on any artificial life model (no Neural Networks or Fuzzy Logic) and is based purely on heuristics-based technology. It does not have any rules and relies entirely on context and feedback. Since it has been online from 1997, it has learned new languages as well.[6]

C. A.L.I.C.E.

A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) is a popular free chatbot developed in 1995 by Dr. Richard Wallace. It is inspired by ELIZA and won the Loebner prize in January 2000. It was first implemented in SETL, a language based on set theory and mathematical logic. This was known as Program A. There were a few contributors at this time until 1998 when it migrated to Java. Dr. Wallace also implemented AIML (Artificial Intelligence Markup Language), an XML dialect for creating chatbots. A.L.I.C.E was written in AIML. In 1999, a Program B was launched where more than 300 developers joined the effort to write A.L.I.C.E and AIML made a transition to fully-compliant

XML grammar. The Program B led A.L.I.C.E to win the Loebner prize in 2000. Program C was launched after Jacco Bikker created the first C and C++ based implementation of AIML. This led the way for many different C/C++ threads for the Alicebot engine. Finally, Program D led the transition of A.L.I.C.E from pre-Java to add many features and to make use of features like Swing and Collections.[5]

D. Commercial Chatbots

The bot SmarterChild was launched in 2001[13] as a service which could answer fact-based questions like "What is the population of Indonesia?" or more personalized questions like "What movies are playing near me tonight?". Before being taken down in 2008, it was a popular chatbot used by thousands of people every day. IBM Watson was developed in 2006 specifically to compete with the champions of the game of Jeopardy! Watson won the competition in 2011. Since then, IBM Watson offers services to build chatbots for various domains which can process large amounts of data. IBM calls this "Cognitive Computing", claiming to use techniques used by humans for understanding data and reasoning. The launch of Siri in 2010 as an intelligent virtual assistant paved the way for other virtual assistants like Google Now (2012), Cortana (2015) and Alexa (2015). All of these assistants can help answer questions based on the web and help access a mobile device easily.[6]

Table 1: Survey of Design techniques of Chatbots

Year	Programme Name	Winner Designer Name	Design Technique
1991	PC Therapist	Joseph Weintraub	Canned and non-sequitur responses in addition to pattern matching after parsing, and word vocabulary that make it remember sentences.
1994	TIPS	Thomas Whalen	A personal history model database like the system with pattern matching.
1995	PC Therapist	Joseph Weintraub	The same as in 1991.
1996	HeX	Jason Hutchens	Has got a trick sentences database, Markov Chain models, pattern matching, and a model of personal history.
1997	Converse	David Levy	A database for facts, pattern matching, proactivity, WordNet synonyms, a statistical parser, ontology, a list of proper names, and a modular of weighted modules.
1998	Albert One	Robby Garner	Hierarchical structure of previous Chatbots, such as Fred, Eliza, pattern matching and proactivity.
2000	A.L.I.C.E	Richard Wallace	Advance pattern matching, AIML.
2002	Ella	Kevin Copple	Language tricks, phrase normalisation, pattern matching, WordNet, and expanding abbreviation.
2003	Jabberwock	Juergen Pirner	Markov Chains, simple pattern matching, context free grammar (CFG), and parser.
2004	A.L.I.C.E	Richard Wallace	The same as in 2000.
2005	George (Jabberwacky)	Rollo Carpenter	No scripts or pattern matching, a huge database of responses of people, and they are based on the Chatbot Jabberwacky.
2007	UltraHAL	Robert Medeksza	Scripts of pattern matching and VB code combination.
2008	Elbot	Fred Roberts	Commercial Natural Language Interaction system.
2009	Do-Much-More	David Levy	Intelligent Toys Commercial Property.
2010	Suzette	Bruce Wilcox	AIML based chat script with database of variables, triples and concepts.
2012	Chip Vivant	Mohan Embar	Responses using unformatted chat script and AI, and ontology.
2013	Mitsuku	Steve Worswick	Based on rules written in AIML [17].
2014	Rose	Bruce Wilcox	It contains a comprehensive natural language engine to recognize the meaning of the input sentence accurately. A chat script is also included in the design

The Survey of the Design Techniques of Chatbots is presented in Table 1 below.

III. DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

A.DEEP LEARNING

Many of the latest deep learning techniques have demonstrated promising results across different kinds of applications Natural Language Processing (NLP) is one of them. Deep learning uses multiple layers to represent the abstractions of data to build computational models. Training a DNN is an optimization process, i.e., finding the parameters in the network that minimize the loss function. In this section, popular deep learning networks such as RNN, LSTM model used in NLP is discussed further. The long term dependency issues of RNN and also the single sequence processing were resolved using Sequence to Sequence Translation model. This model processes two sequences both input and output. Recurrent Neural Network (RNN)

RNN is most popular algorithm in deep learning, used in NLP and speech processing. Unlike traditional neural networks, RNN utilizes the sequential information in the network. This property is essential in many retrieval based applications and in the embedded structure with feedback that conveys useful knowledge. For example, to understand a word in a sentence, it is necessary to know the context. Therefore, an RNN can be seen as short-term memory units that include the input layer x, hidden (state) layer s, and output layer y. RNN helps in reducing the difficult learning in deep networks.

Recurrent neural networks (RNNs) are specialized models for sequential data. Recurrent Neural Network (RNN) are known to preserve the state of previous neuron. This approach makes RNNs desirable for chatbots.[2] Chatbots retaining context in a conversation is essential to understand the user. The RNN can easily map sequences to sequences whenever the alignment between the inputs the outputs is known ahead of time.[1] RNNs perform well in the task of speech processing. Later researchers found that RNN does not work well with long input sequences. The major concern with the standard RNN here was it could not learn the long dependencies. To relieve this drawback LSTMs were used. [3]

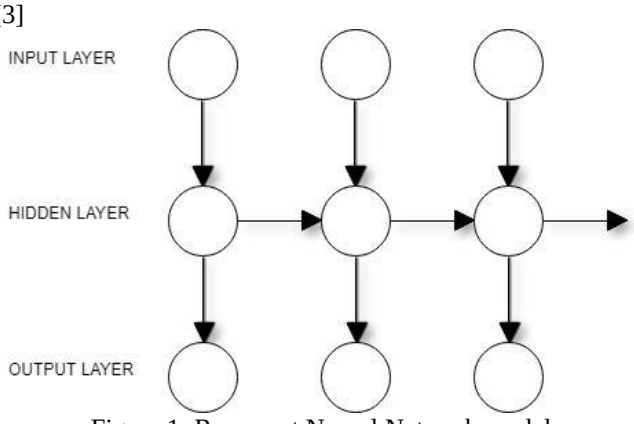

Figure 1: Recurrent Neural Network model

Long-Short Term Memory (LSTM)

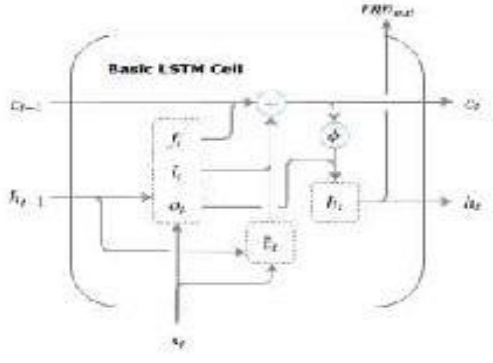

Figure 2: Basic LSTM Cell

Long Short-Term Memory (LSTM) is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNN. [1] In Deep Neural Networks with deeper architectures, deep LSTM RNNs have been successfully used for speech recognition. [] The LSTM contains a special memory unit in the recurrent hidden layer. These memory blocks contain memory cells with self-connections storing the temporal state of the network. In addition there are special multiplicative units called gates. These gates control the flow of information. Each memory block in the original architecture contained an input gate and an output gate. The input gate controls the flow of input activations. The output gate controls the output flow of cell activations into the rest of the network. Sequence to Sequence model RNNs can be used as language models for predicting future elements of a sequence given prior elements of the sequence. However, some necessary components for building translation models are missing as we can operate only on a single sequence, while translation operates on two sequences the input sequence and the translated sequence. Sequence to sequence models build on top of language models consist of an encoder step and a decoder step. In the encoder step, a model converts an input sequence into a fixed representation. In the decoder step, a language model is trained on both the output sequence (such as the translated sentence) as well as the fixed representation from the encoder. Thus the language model now allows modeling of simple sequences by predicting the next word in a sequence, given a previous word in the sequence. This makes the translation process easier and faster at the same time.

B. NATURAL LANGUAGE PROCESSING (NLP)

Natural language processing, in its simplest form, is the ability for a computer/system to truly understand human language and process it in the same way that a human does. [1] Speech is a complex time-varying signal with complex correlations at a range of different timescales.[1 LSTM]

There are various logical steps involved in Natural language Processing which are as follows:

- Phonetics/phonology: The study of linguistic sounds and their relations to written words.
- Morphology: The study of internal structures of words/composition of words.
- Syntax: The study of the structural relationships among words in a sentence.
- Semantics: The study of the meaning of words and how these combine to form the meaning of sentences.
- Pragmatics: Situational use of language sentences.
- Discourse: A linguistic unit that is larger than a single sentence (context)

The Figure below shows the Logical Processes involved in Natural Language Processing.

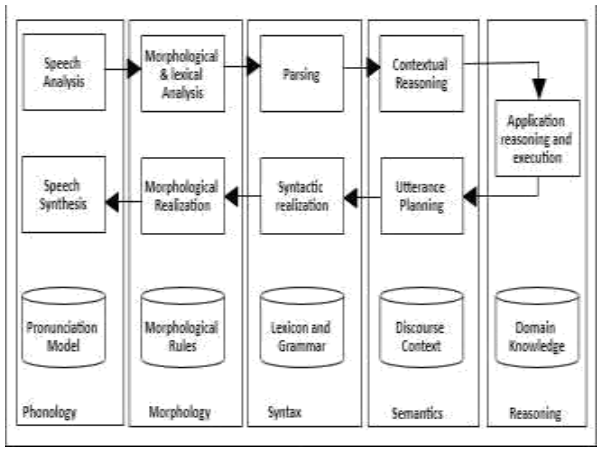

Figure 3: Logical steps involved in Natural language Processing

The basic processes involved in text processing are explained below.

A. Tokenization

Tokenization is the initial step in NLP. Also we need to understand the Natural Languages are not well formed or have well defined structure as in textbook or newspapers. Text processing of natural language is quite complex due to variety of languages and the accents. Text processing can be divided into 2 steps: Document triviage and Text Segmentation. Document Triviage is the process of converting the files in a document into well-defined text document. While Text Segmentation is the process of converting well defined corpus text into meaningful units known as 'Tokens'. Text Segmentation can also be referred to as Tokenization. Text Normalization is another step in text-processing that converts these tokens into it's canonical form. many dependencies are to be considered during text segmentation such as Character set dependency, Corpus Dependency, Language Dependency, Application Dependency, etc.

B. Lexical Analysis

Words being the building blocks of the NLP, it's important to understand the morphology of words occurring in a text. Sometimes the words in the text tend to be morphologically complex. The analysis performed at the level of a word is lexical analysis. A word can be thought of as a string or an abstract object. A basic task for which lexical analysis is performed is for relating the morphological variants of a word to their lemma. This lemma lies in a lemma dictionary. Lemma dictionary is bundled up with the invariants of word with its semantic and syntactic information. While Mapping a word variant to lemma dictionary is one side of the Lexical Analysis, parsing is other. Various models for Lexical Analysis are FSM(Finite State Morphology), Paradigm based and so on. The FSM model is used because of its computational Efficiency and invertibility. FSM can be used for both generation and parsing of morphologically complex structure.

C. Syntactic Parsing
Parsing is breaking down the sentence into its constituent words. A sentence is parsed to find out the grammatical type of the words. Parsing of computer languages is easy, not that's not the case with natural languages. Parsing in NLP means to analyze the input sentence, identify the parts of speech of the grammar constituents. Natural Language processing uses two parsing techniques viz. Top-Down and Bottom-Up. The name itself describes the direction in which parsing process advances. Mostly used parsing algorithms are Context-Free Grammar(CFG) and Top-Down parser. CFG is the grammar that consists of rules with a single symbol on the left-hand side of the rewrite rules. In Top-Down parsing technique, parser searches for a parse tree by trying to build from the root node down to the leaves. Bottom – Up parsing starts with the words from input and tries to build trees from the words up, again by applying rules from the grammar.[2]

Semantic Analysis
Generally in linguistics, semantic analysis refers to analyzing the meanings of words, fixed expressions, whole sentences, and utterances in context. In practice, semantic analysis means translating original expressions into some kind of semantic meta language. The semantic analysis of natural language content is to capture the real meaning of any text. It identifies the text elements and assigns them to their logical and grammatical role. It analyzes context in the surrounding text and it analyzes the text structure to accurately disambiguate the proper meaning of words that have more than one definition.

E. Text Generation
Prior to the last few years, the predominant techniques for text generation were either based on template or rule-based systems, or well-understood probabilistic models such as n-gram or loglinear models. Natural language Generation entails generating documents or longer descriptions from structured data.

IV. LINGUISTIC MODEL

A. Word2Vec Model
In recent two years the word2vec model & application by Mikolov et al. have attracted a great amount of attention. The vector representations of words learned by word2vec models have been shown to carry semantic meanings and are useful in various NLP tasks.[3] Word2vec is a "predictive" model. An efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words is provided by this tool. Natural language processing applications use these representations subsequently. The input to the word2Vec tool is given through a text corpus and it produces the word vectors as output. Initially a vocabulary is constructed from the training text data and then it learns vector representation of words. These word vector file can be used as features in many natural language processing and machine learning applications. A simple way to find the learned representations is to find the closest words for a user-specified word. The two main learning algorithms in word2vec are : Continuous Bag-of-Words (CBOW) and Continuous Skip-gram. Both of these algorithms learn the representation of a word that is useful for prediction of other words in the sentence.

B. The Skip-gram Model
The Skip-gram model's training objective is to find word representations that are useful for predicting the surrounding words in a sentence or a document. Recently, Mikolov et al.[3] introduced the Skip-gram model, an efficient method for learning high quality vector representations of words from large amounts of unstructured text data. It does not involve dense matrix multiplication. Thus making the training extremely efficient: an optimized single-machine implementation can train on more than 100 billion words in one day.

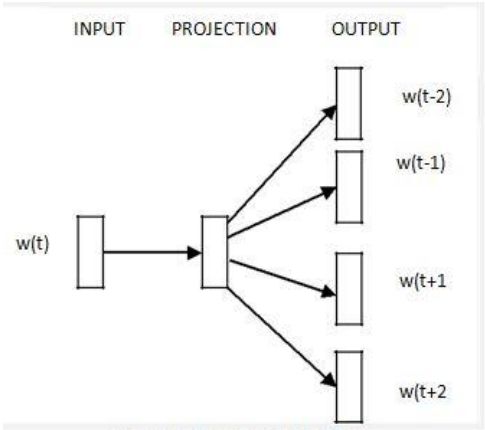

Figure 4: Skip-Gram Model []

Continuous Bag of Words (CBOW)

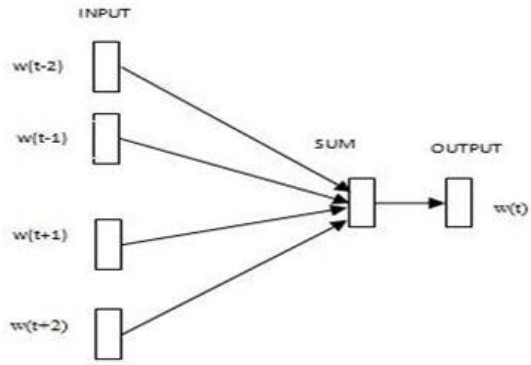

Figure 4: CBOW Model []

Mikolov et. al in 2013 presented an architecture .i.e. Continuous Bag of Word. This model considered all words are independent of each other. Unlike standard bag-of-words model, it uses continuous distributed representation of the context. The Word2Vec family of models are unsupervised and thus the generation of dense word embeddings need just a large corpus, be the data be unlabelled.

C. Hidden Markov Model
An HMM is a probability distribution over a sequence of observations.[4] This model is defined by two properties. First, it assumes that the observation generated at time t by some process whose state is hidden from the observer. Hidden Markov models (HMM) is a probabilistic model

consisting of variables representing observations, hidden variables, the initial state distribution, a transition matrix, and parameters for all observation distributions.[1] Second, the Markov assumption is that the probability of an output observation depends only on the state that produced the observation and not on any other states or any other observations.[1, 2]

HMM uses the Markov Chains. A Markov chain is a model that tells us something about the probabilities of sequences of random variables, states, each of which can take on values from some set. These sets can be words, or tags, or symbols representing anything, like the weather.[3] A Markov model embodies the Markov assumption that when predicting the future, the past doesn't matter, only the present matters. It is well known for its efficiency in modeling short-term dependencies between adjacent symbols. The palindrome language, the language that contains all strings that read the same forward and backward, cannot be represented using regular grammars - or equivalently, using HMMs.[2]

The csHMM introduced by Byung-Jun Yoon et al. is a context-sensitive HMM that is capable of modeling long-range correlations, by rendering certain states in the model context-sensitive. The csHMM is also a doubly stochastic process, which consists of a non-observable process of hidden states and a process of observable symbols. HMM has been widely used in digital communication applications such as signal processing.

D. GLOVE Model

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training takes place by aggregation of global word-word co-occurrence statistics from a corpus. The resulting representations are the linear substructures of the word vector space. GloVe is a global log-bilinear regression model with a weighted least-squares objective. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. The performance analysis of the word analogy dataset gives 75% accuracy. There are various modified versions of the basic models explained above. These models are N-grams, Bag of Words, etc. Every NLP tool needs to be evaluated to check the performance. The performance evaluation Techniques are explained in the section below.

V. EVALUATION OF LANGUAGE MODELS

Daniel et al states the best evaluation practice is to embed the model into the application and measure the improvement of application. There are two types of evaluation: Extrinsic Evaluation and Intrinsic Evaluation. Extrinsic Evaluation is an end-to-end evaluation. Running NLP in big system is often very expensive so in that case an intrinsic evaluation is preferred which is independent of application. An intrinsic Evaluation simply needs a test set. In the fit test for two different models, the higher probability model is considered to produce more accurate results and hence is a better model. This is when the evaluation is based on the test set probability. This type of testing set introduces bias making all probabilities high. This produces inaccuracies in perplexity. Perplexity is a variant used instead of raw probabilities for evaluation of Language models. PP(W) the perplexity of word is given below in equation (i)

$$PP(W) = \left(\left(\frac{1}{P(w_1\, w_2 \ldots w_N)}\right)\right)^{\frac{1}{N}} \quad \ldots.(i)$$

VI. RELATED WORK

Alan Turing in 1950 propose to consider the question, "Can Machines Think?" Turing put forward the idea of an 'imitation game', in which a human being and a computer would be interrogated under conditions where the interrogator would not know which was which. The communication entirely being by textual messages. Turing argued that if the interrogator could not distinguish them by questioning, then it would be unreasonable not to call the computer intelligent, because we judge other people's intelligence from external observation in just this way.[1] Joseph Weizenbaum in 1966 created a computer based interface that would communicate with humans. Most popularly known as ELIZA the first computer program to pass the Turing Test. ELIZA used keyword detection technique to generate responses. When no keyword would match ELIZA would generate some random out of context responses.[4] After ELIZA various Chatbots were build like A.L.I.C.E. winner of the Loebner prize in January 2000.[5] A.L.I.C.E. being the first program implemented using C and C++ led to evolution of further chatbot implementation using programming language. IBM Watson was developed in 2006 specifically to compete with the champions of the game. Siri's launch in 2010 paved a way for other Virtual assistants based on voice. Later Google Assistant(2012), Cortana(2015), Alexa(2015) were implemented using similar technologies. With the AIML based chatbots the implementation language models got easier. The retrieval based models used RNN model along with LSTM. These models had memory unit and hence were used to store responses for the chatbot. The deep Learning made the task easier with its wide range of application in the field of Natural Language processing.[5] Deep Learning models such as Sequence to Sequence models overcome the drawbacks of the Long term dependencies and also it was used for translation purpose. The sequence to Sequence model use in uses two LSTMs one for encoder and one for Decoder. The Drawback of the sequence to sequence model was it introduced short term dependencies. This drawback was overcome by Clustering based Adaptive Sequence to Sequence Learning model in [18] by Sutskever et. al. This model used the clustering techniques to avoid extra computation. Adaptive clustering model use several Seq2seq models to train different clusters of data. This clustering helped to sort the text based on categories. The Evaluation of these Language models can be done using the BLUE: a method for Automatic evaluation of machine translation. The BLUE score is generated by comparing the n-grams of candidate with the reference translation and thereby count the number of matches. Moving further from simply

predicting the sequences and the context using sequential models and n-grams Nirmala et. al. in [] proposed a system that uses tensorflow to implement a neural network and trained it with intent file to generate a response model. The accuracy of this Chatbot is directly proportional to the size of intent file used for training the chatbot. Thus the larger the training set the better the performance of the model.

VII. CONCLUSION

In this paper we presented the overview of Natural Language Processing. We studied all the basic techniques that are essential in building a chatbot. The models for text processing such as tokenization, parsing, text segmentation, text normalization and so on are studied. After studying various Deep Learning models for Natural Language processing we conclude that the Recurrent Neural Network is best suitable for building Chatbot over Convolutional Neural network. But we also come to a conclusion that LSTMs should be preferred to overcome the Long term dependencies. As presented above LSTM have an additional feature .i.e. the gates of LSTM. Also using number of LSTMs we can characterize the text into different clusters which is another advantage over the traditional sequence to sequence model. Based on the pros and cons of the above stated techniques and modalities we will implement a Chatbot in a specialized Domain.

REFERENCES

[1] Turing, Alan M. "Computing Machinery and Intelligence." Creative Computing 6.1 (1980): 44-53.
[2] Smestad, Tuva Lunde. Personality Matters! Improving The User Experience of Chatbot Interfaces-Personality provides a stable pattern to guide the design and behaviour of conversational agents. MS thesis. NTNU, 2018.
[3] Bhagwat, Vyas Ajay. "Deep Learning for Chatbots." (2018).
[4] Weizenbaum, Joseph. "ELIZA—a computer program for the study of natural language communication between man and machine." Communications of the ACM 9.1 (1966): 36-45.
[5] Wallace, Richard S. "The anatomy of ALICE." Parsing the Turing Test. Springer, Dordrecht, 2009. 181-210.
[6] Carpenter, Rollo. "Jabberwacky." (2007).
[7] Abdul-Kader, Sameera A., and J. C. Woods. "Survey on chatbot design techniques in speech conversation systems." International Journal of Advanced Computer Science and Applications 6.7 (2015).
[8] V. Bhargava, and N. Maheshwari, "An Intelligent Speech Recognition System for Education System," 2009.
[9] J. Ratkiewicz, "Evolutionary Sentence Combination for Chatterbots Dana Vrajitoru Computer and Information Sciences Indiana University South Bend, 1700 Mishawaka Ave," 2004.
[10] M. J. Pereira, and L. Coheur, "Just. Chat-a platform for processing information to be used in chatbots," 2013.
[11] P. Van Rosmalen, J. Eikelboom, E. Bloemers, K. Van Winzum, and P. Spronck, "Towards a Game-Chatbot: Extending the Interaction in Serious Games," 2012.
[12] A. S. Lokman, and J. M. Zain, "An architectural design of Virtual Dietitian (ViDi) for diabetic patients." pp. 408-411, 2009.
[13] A. S. Lokman, and J. M. Zain, "Extension and prerequisite: An algorithm to enable relations between responses in chatbot technology," Journal of Computer Science, vol. 6, no. 10, pp. 1212, 2010.
[14] A.S. Lokman, and J. M. Zain, "One-Match and All-Match Categories for Keywords Matching in Chatbot," American Journal of Applied Sciences, vol. 7, no. 10, pp. 1406, 2010.
[15] F. A. Mikic, J. C. Burguillo, M. Llamas, D. A. Rodríguez, and E. Rodríguez, "CHARLIE: An AIML-based Chatterbot which Works as an Interface among INES and Humans." pp. 1-6, 2009.
[16] D. Vrajitoru, "Evolutionary sentence building for chatterbots." pp. 315-321, 2003.
[17] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.
[18] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.

