Spring 5-20-2019

Chatbots with Personality Using Deep Learning

Susmit Gaikwad
San Jose State University

Chatbots with Personality Using Deep Learning

A Thesis

Presented to

Prof. Robert Chun

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Class

CS298

By

Susmit Gaikwad

May 2019

The Designated Thesis Committee Approves the Thesis Titled

Chatbots with Personality Using Deep Learning

By

Susmit Gaikwad

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2019

Dr. Robert Chun, Department of Computer Science

Dr. Katerina Potika, Department of Computer Science

Mr. Vyas Bhagwat, Wave Computing

ABSTRACT

Natural Language Processing (NLP) requires the computational modelling of the complex relationships of the syntax and semantics of a language. While traditional machine learning methods are used to solve NLP problems, they cannot imitate the human ability for language comprehension. With the growth in deep learning, these complexities within NLP are easier to model, and be used to build many computer applications. A particular example of this is a chatbot, where a human user has a conversation with a computer program, that generates responses based on the user's input. In this project, we study the methods used in building chatbots, what they lack and what can be improved.

Index Terms – **Chatbots, deep learning, natural language processing, neural networks**

TABLE OF CONTENTS

I. INTRODUCTION

Although technology has changed the way how humans interact with devices and digital systems, accessibility is still largely limited. Chatbots allow for seamless interaction between software systems and humans through natural language processing, giving users an illusion of talking to another actual person. Existing chatbot models built into digital assistants like Alexa, Cortana and Siri are used to make human lives easier by providing ease of interaction and easy information retrieval. But these systems often have trouble determining the intentions of the user. Specifically, in a conversation they fail to follow up on context. For example, if users told Siri to make a note of something they said, she would actually create a digital note. In this context, the phrase means *'to make a mental note for future reference'* and is taken out of context.

The Turing Test, created by Alan Turing in 1950, was designed to test a machine's "intelligence" to determine whether a person could distinguish it from a human or not. Since then, there have been many attempts to make a chatbot that could clear the test [7]. In 1966, a program called ELIZA that matched keywords from a user inputted text, cleared the Turing Test but only partially [11]. If ELIZA found the keyword in its dictionary it gave an appropriate response, otherwise it gave a generic message. This worked for a search-based scenario but failed in conversation-based interactions. Ruled-based chatbots have been tried and tested but they fail to acknowledge the randomness in conversations and cannot change context according to how the user's context changes. In 1972, psychiatrist Kenneth Colby designed PARRY, which simulated the thinking of a paranoid person or schizophrenic. PARRY consistently misinterpreted what people said and assumed they had nefarious motives and that they were always lying. This was the first time any sort of personality was implemented in a chatbot. It also became the first machine to pass a version of the Turing Test.

Since chatbots are supposed to imitate human beings, methods that use Artificial Intelligence (AI) are incorporated to build them. One such method is deep learning, which is based on how the human brain works. Deep learning models have different architectures, and these can be application specific. Deep learning models i.e. Artificial Neural Networks (ANNs), discover patterns hidden in the training data and model the new data to fit within these patterns. Deep learning is giving impressive results when used to solve problems in the field of Computer Vision and Natural Language Processing (NLP). This is supported by the fact that Google has invested in AI startups through the Google Assistant Investment Program to broaden the Assistant's set of features, build new hardware devices for digital assistants, and/or focus on a particular industry.

This project explores various ANN architectures like Recurrent Neural Networks (RNN). RNNs are deep learning architectures designed to handle sequential data as they are used in chatbots. The project attempts to gain insight into the following questions: *What is Deep Learning? Why is a chatbot needed? How are chatbots built currently? What limitations exist and how can they be overcome?* Articles crucial to this problem will be reviewed along with recent developments in this space. These articles involve Conference papers, books, Journals and technical reports. Fig. 1 is a conceptual outline of the topics covered in this paper.

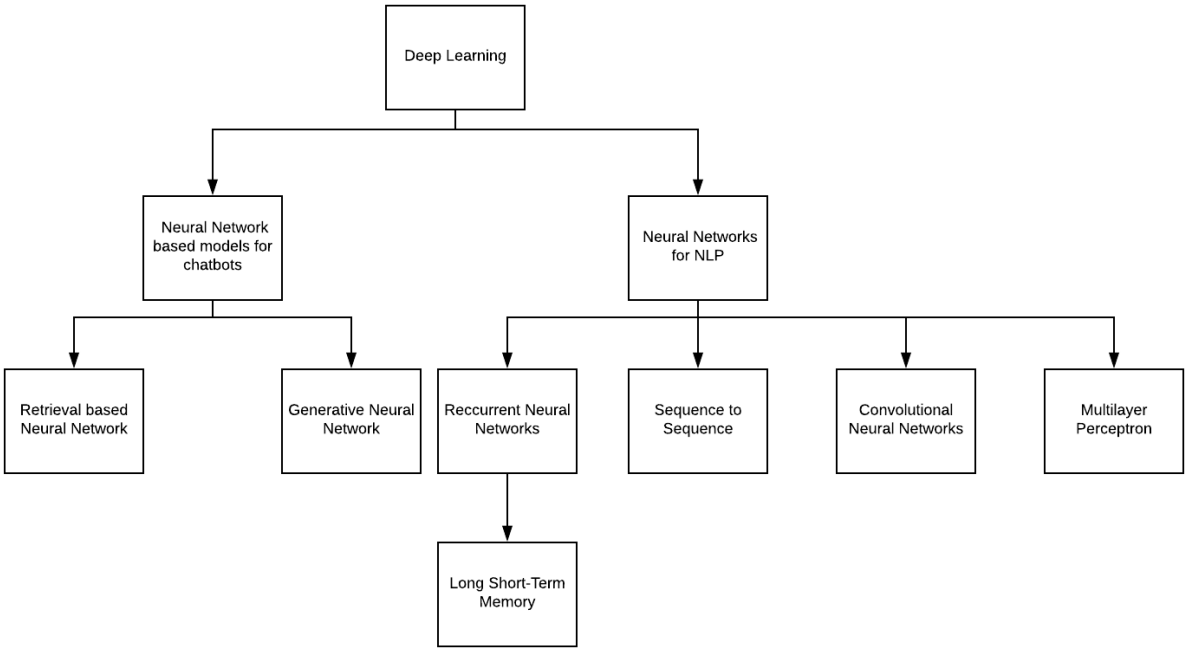


Fig. 1. Conceptual oultine of topics

.

II. THE BASICS OF DEEP LEARNING

Deep learning architectures are based on Artificial Neural Networks (ANN) but with more than one hidden layer. An ANN models the human brain, with the ANN computing units analogous to neurons in the brain, and weights analogous to the connection strength between neurons. The multiple hidden layers in deep learning allow us to model complex functions of the brain, like differentiation and recognition of patterns and objects. With the increase of research in deep learning, there is a growth in models that are able to carry out complex, intellectual tasks like humans such as identifying objects in images, creating music and holding conversations. The multiple layers of data-exchanging neurons allow it to learn denser representations of data with deep levels of abstraction [18].

Using Deep learning we can train a model end-to-end for the purpose of an application. This is because the models offer deeper representation of patterns in the data, and this information can be 'encoded' into the network. In machine translation, the network is constructed from a translated corpus, which involves pairs of sentences in two languages and usually no human intervention is needed. In most machine translation approaches in statistics, feature engineering is very crucial [17] [18]. In deep learning, the representations of data in different forms, can be learned as distributed vectors. This allows for information processing across multiple modality.

The real reason behind using deep learning for chatbots is two-fold: the availability of data and computational resources [17]. Neural networks have been around since a long time but it's only now that we have access to huge amounts of data and powerful computational systems like Graphics Processing Units, which facilitate deep learning. While other machine learning methods

perform NLP tasks fairly well, when given huge amounts of data deep learning models outperform them all. Architectures like Recurrent Neural Networks (RNN) take advantage of the sequential nature of language and speech to learn underlying patterns to create well-crafted responses to user inputs [15].

III. HISTORY OF CHATBOTS

A. ELIZA

First coined as “Chatterbot”, the term was coined in 1994 by Michael Mauldin to define conversational systems that attempted to pass the Turing Test. ELIZA, created in 1966 by Joseph Weizenbaum is known for being the first conversational system to pass the Turing Test partially. ELIZA’s model understands the user’s input and tries to match the keywords with pre-defined responses. ELIZA’s design was such that conversed through pattern matching and substitution. ELIZA’s program was made to understand patterns of human communication and respond with answers that involved the similar substitutions. This gave the illusion that context of the conversation was understood by ELIZA [11].

Consider an example. In the input sentence 'I want to run away from my parents', ELIZA assigns a weighted value to every word in the sentence. Low values are to pronouns, higher values to action verbs, and the highest value to the actual action. This allows the program to know exactly how to flip the sentence around to ask a digging question. ELIZA recognizes the particular rule associated with the action verb, then responds with the appropriate matched response. The reply given then is, "What would getting to run away from your parents mean to you?”. Despite the rule matching system, ELIZA had some responses which were generic for any context, probably for when no rule matching was found. For example,

Input: I had a great day!

ELIZA: Please go on.

ELIZA was very popular as people thought they were talking to a real therapist. This was contrary to the fact that Weizenbaum created ELIZA to show the superficiality of conversation

between a human and a machine. ELIZA being rule-based couldn't hold meaningful conversations with people and had generic replies to inputs which did not comply with any rules. Many other chatbot implementations were inspired by this simple rule-based design, like PARRY and A.L.I.C.E.

B. PARRY

PARRY, created in 1972 by psychiatrist Kenneth Colby, tries to simulate the conversational model of person with paranoid schizophrenia [12]. PARRY implemented a crude model of the behavior of a paranoid, schizophrenic person. This behavior was based on judgements and beliefs about conceptualizations on reactions of accept, reject or neutral. It was a much more complicated and advanced program than ELIZA, embodying a conversational strategy. PARRY was evaluated using a version of the Turing Test in the 1970s. Experienced psychiatrists analyzed a mix of actual patients and computers by showing PARRY's response on teleprinters. 33 different psychiatrists were then shown transcripts of the conversations and were asked to identify which of the "patients" were computer programs and which were actually human. They were able the correct guess only 48 percent of the time — which is as good as a random guess.

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

A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) is a chatbot created in 1995 by Dr. Richard Wallace in AIML (Artificial Intelligence Markup Language), an XML dialect for creating chatbots. It is inspired by ELIZA and won the Loebner prize in January 2000 [13]. The first implementation was called Program A. Program A was written in a language established on concepts of set theory and mathematical logic called SETL. In 1998 it was rewritten in Java. Program B was implemented in 1999, when 300 software developers came together to write

A.L.I.C.E and AIML was changed grammatically to be fully-compliant to XML. A.L.I.C.E won the Loebner prize, as Program B in 2000. In 2000, Program C was created as a C/ C++ based implementation of AIML by Jacco Bikker. In November 2000, Program D was implemented as a bit named DANY by Jon Baer. Program B refactored with new features using Java 2 was implemented as Program D and this implementation was supported by the A.L.I.C.E. AI Foundation. Developments in the AIML community made way for the creation of Program E, an AIML interpreter written in PHP.

D. Jabberwacky

Jabberwacky is a chatbot created by Rollo Carpenter, a British programmer. It is meant to be a conversational bot to mimic natural human interaction for entertainment purposes [14]. It was one of the first attempted chatbots to implement AI to simulate meaningful human conversations. It won the Loebner prize in 2005 and 2006, an annual competition to test chatbots. Jabberwacky was also able to learn continually during the conversation. The makers have declared that Jabberwacky is not implemented using any artificial life model (no Neural Networks or Fuzzy Pattern Matching) and is based purely on heuristics-based algorithms. It does not have any rules hard-coded, instead relying on feedback. Jabberwacky stores everything the users have inputted and determines the most appropriate response using contextual pattern matching techniques. Hosted online since 1997, Jabberwacky has also managed to master new languages.

E. Commercial Chatbots

SmarterChild was the first commercial chatbot to get widespread adoption. Launched in 2001 by ActiveBuddy as a web service integrated in AOL's Instant Messenger [15]. It gave answers to fact-based questions like "What is the capital of Italy?" or more questions like "Who

won the Lakers game last night?". Before its demise in 2007, it enjoyed record growth and user engagement as a 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. Recently, they have enhanced capabilities like Image-based search. Since all of these technologies are commercial, the details of implementation are not available.

IV. TYPES OF MODELS FOR CHATBOTS

A. RETRIEVAL-BASED MODEL

These types of models give responses based on a repository of predefined responses. Based on the input and context from user, a certain heuristic is defined to select the appropriate response. This heuristic could be as simple as a rule-based expression match, or as complex as an ensemble of Machine Learning classifiers. Here no new text is being generated, instead it just returns the most likely response based on the responses it already knows. Due to this they are unable to handle unseen data. As they do not generate new responses there are no grammatical errors either. P. Goyal, et al., discuss both models and how a rule-based system can be used for menu driven chatbots [4]. I. Thies et al., mention retrieval-based models and developing them using deep learning [8].

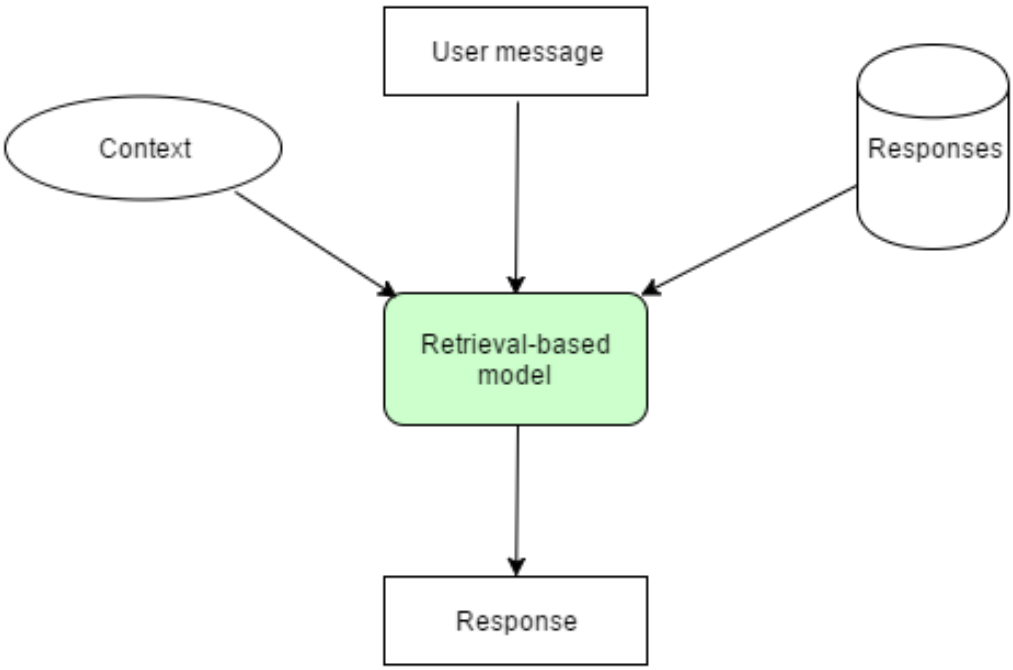


Fig. 2. Retrieval-based system

B. GENERATIVE-BASED MODEL

Generative models do not use responses that are already defined, instead they create new responses and generate new text. Generative based models take inspiration from Machine Translation techniques, but here instead of translating, it learns the user input and output response in the form of translations from input to output. These models tend to be harder to train due to the huge amounts of training data required. They also tend to make grammatical mistakes [4].

Xu et al, I. Thies et al, M. Qiu et al, and S. Mishra et al [6][8][9][10] all agree that the future of chatbots and conversational agents are generative-based models.

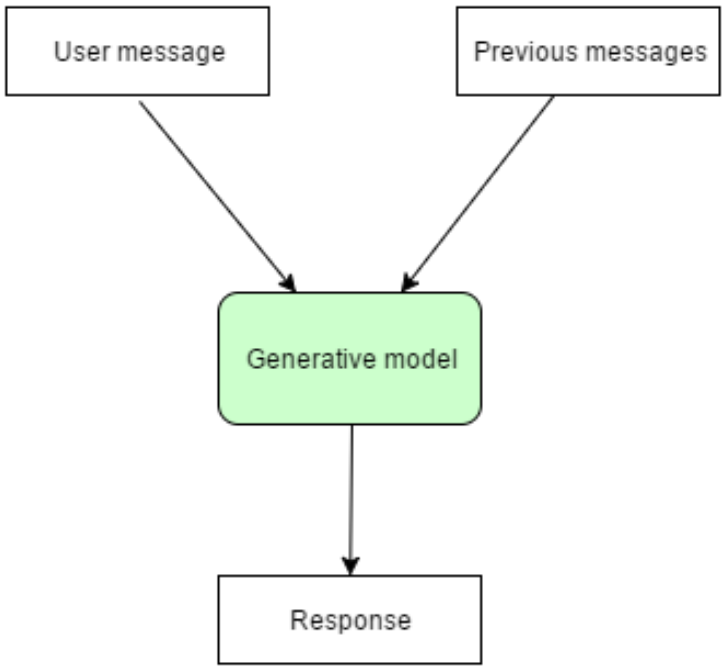


Fig. 2. Generative-based system

V. NATURAL LANGUAGE PROCESSING (NLP) USING DEEP LEARNING

For NLP tasks in deep learning, the input to models are in the form of sentences given in the form of vectors. For every row of the vector, an element refers to a token. Usually tokenizing is defined as using separate words, but sometimes character level tokenizing is done. These vectors represent word embeddings of each sentence (probability or frequency distributions) but sometimes are an index for the word in a dictionary (encoding).

A. MULTILAYER PERCEPTRON (MLP)

An MLP is a neural network with one input and output layer, while having at least one (usually more) hidden layer. MLPs are usually used to classify non-linear data using activation functions like logistic sigmoid function or SoftMax function. Every node in a layer in an MLP is connected to every node in the next layer, i.e. an MLP is a fully-connected network. This kind of architecture with hidden layers forms the basis of deep learning. MLPs are used in NLP tasks like speech recognition and machine translation [18].

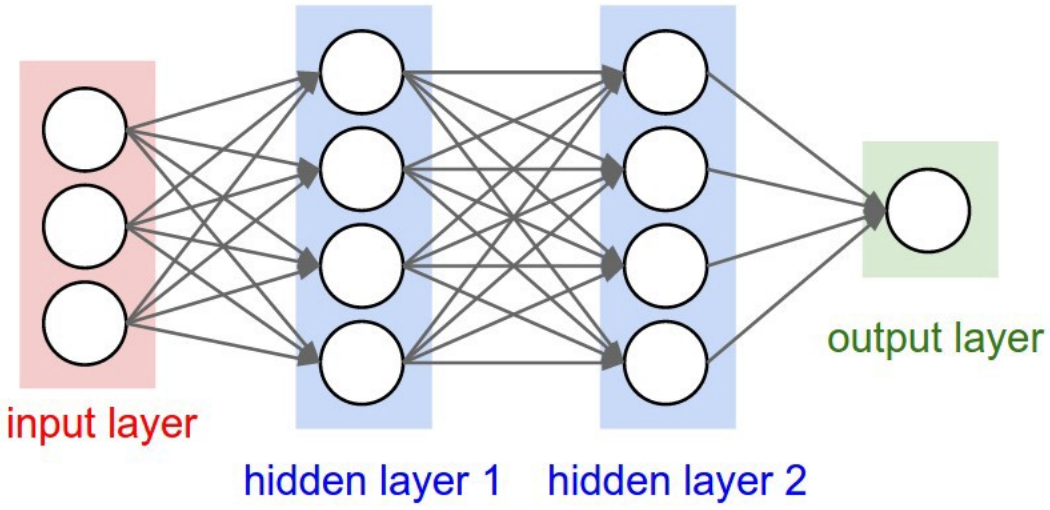


Fig. 3. Multilayer-Perceptron Diagram

B. RECURRENT NEURAL NETWORKS (RNN)

RNNs are deep learning architectures designed to handle sequential data in order to retain the previous data of the neuron. This allows the model to preserve context and give outputs based on previously learned data. An RNN takes advantage of sequential nature of speech and language, every word is encoded with the previous word's information.

A chatbot is required to maintain context to hold a conversation and this factor of RNNs make them a highly desirable choice for NLP tasks like dialogue generation and speech recognition [1][5].

While RNNs seem efficient, there is a problem. At each step we create an understanding of what we read till now, so it's always a mix of the past and the present. When the input is a really long piece of text and the only vital information you really needed to remember was the first word, RNNs fail. They suffer from what is referred to as the long-term dependency problem.

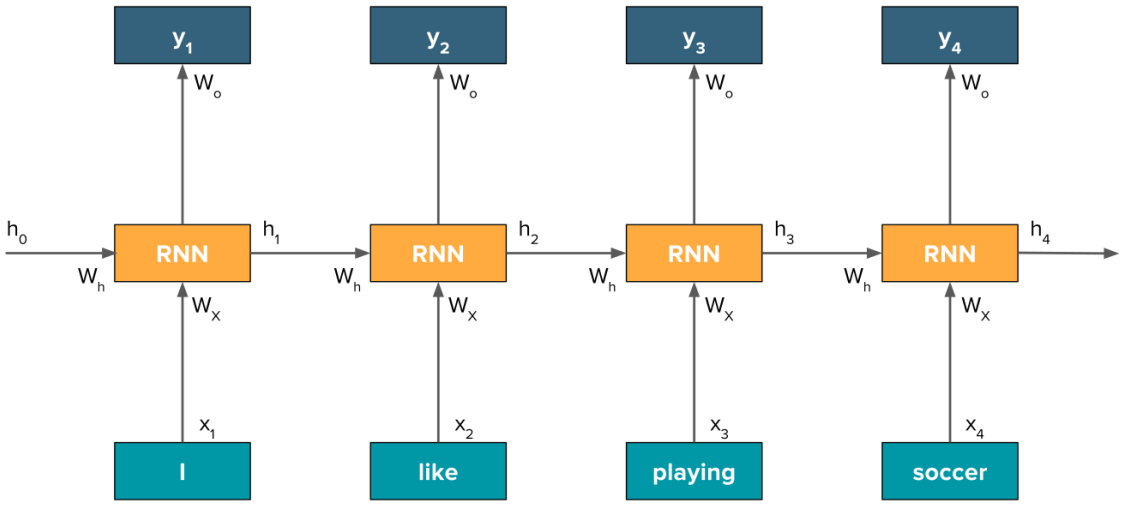


Fig. 4. Input and Output of RNN

Consider the following sentence: "Liverpool had an abysmal performance against Bournemouth on the last weekend and it leads one to believe that the upcoming match at Old

Trafford when they host Arsenal will surely be a difficult one." If asked here, "Which team played against Bournemouth last weekend?", the answer is the very first word. But as part of creating a representation from the sentence, a Vanilla RNN mixes up the embedding for the word "Liverpool" so many times, it will not generate the correct output for the input question. This is because the Vanilla RNN architecture doesn't allow any information in the network to flow unaltered.

1. LONG SHORT-TERM MEMORY (LSTM):

An RNN-based architecture capable of learning long term dependencies via special input, forget and update gates. LSTMs have an additional state called 'cell state' through which the network makes adjustments in the information flow. The advantage of this state is that the model can remember or forget the learnings more selectively. Since LSTMs allow the network to retain the input state for longer, it processes long sequences very efficiently and performs well in terms of accuracy. Le et al in [2] talks about how LSTM models can be used in NLP tasks such as language translation to great success with high accuracy.

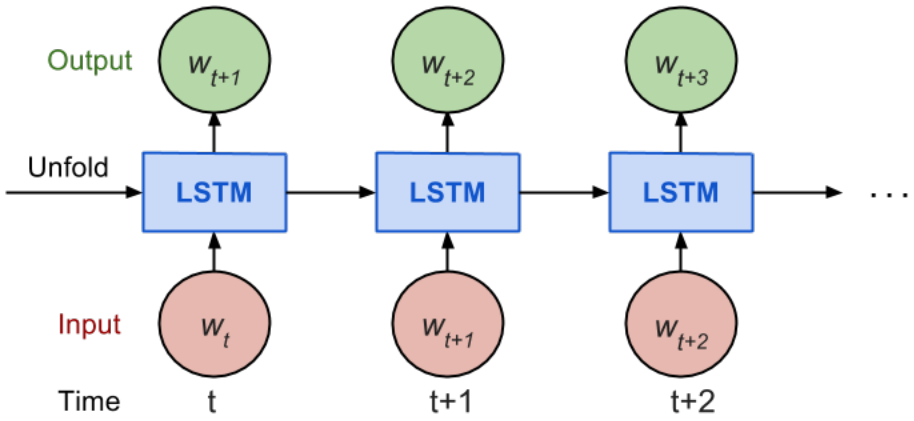


Fig. 5. LSTM cells architecture

C. SEQUENCE TO SEQUENCE (SEQ2SEQ)

Seq2Seq are RNN based models made to tackle problems in sequence to sequence modelling where the input and output are both part of a sequence [3]. A seq2seq model comprises of two components– an encoder and a decoder, both of which are individual networks. The encoder learns the input sequence and outputs a dense, deep representation of that data. The outputs of this network are fed to the decoder, which in turn generates output in the form of a sequence [2].

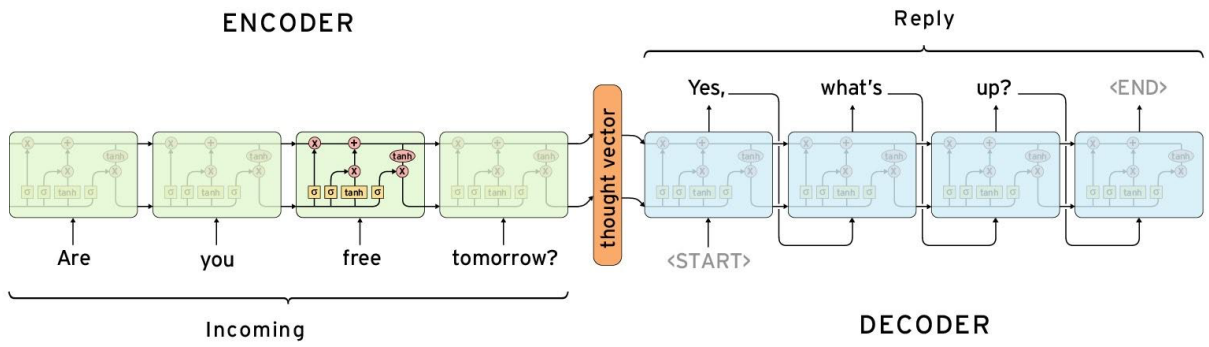


Fig. 6. Seq2Seq Encoder-Decoder Architecture

Seq2Seq models are trained to learn conversations in the form of input-response pairs. It learns the word embeddings for the sentences and thus learns to recognize similar inputs which it learnt and tries to generate the appropriate response [3]. The Seq2Seq architecture has pretty obvious applications in NLP. For example, in chatbots each word in the output response will be generated by the seq2seq model, where the network learns each word of user input text sequentially [2][12]. Lee et al train a Seq2Seq based chatbot [12] on the Twitter chat corpus using Reinforcement learning and get decent results on sentiment coherence, correct grammar and accurate response.

D. CONVOLUTIONAL NEURAL NETWORKS (CNN)

CNNs are primarily MLPs with at least one convolutional layer that reduce the complexity of a network by applying a convolutional function to the input, while analyzing the part of the data at a time before passing the output to the next layer. Since the complexity is effectively reduced, the architecture of the neural network can be denser with more neurons to handle data which is much more complex [13].

Although CNNs are mostly used for computer vision tasks, Collobert et al [14] successfully used convolutional layers as part of an ANN model to do NLP tasks like semantic analysis and speech tagging which act as preprocessing steps for chatbots. They also propose a CNN architecture to perform multiple NLP tasks at once which helped the case of CNNs in NLP based problems.

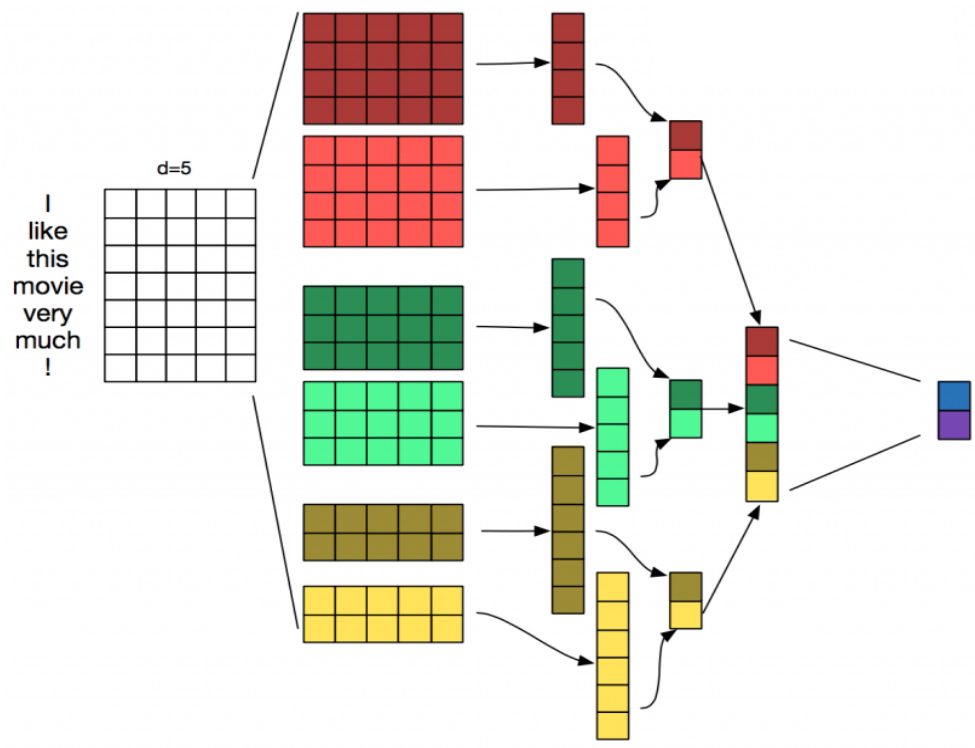


Fig. 7. CNN sentence vector representation

In the context of text, CNNs use convolutional filters that slide over the words. The words in a sentence are represented as a full row in the matrix. Thus, the length of the filter is actually the length of the input matrix. The length might be different, but filters usually slide over 2-5 words at a time. Convolution filters give feature maps for the respective region size which are then used for pooling functions. The concatenation of these individual vectors gives a vector representation of that sentence. It can be further used for classification.

VI. SUMMARY OF CURRENT STATE-OF-THE-ART

Deep learning is an exciting subject with a lot of promise due to the huge amount of research being carried out in this domain, allowing us to analyze large amounts of data in a scalable fashion. It helps eliminate the need for feature engineering as opposed to other traditional machine learning methods and allows the model to truly understand deep lying data patterns.

Deep learning architectures like Seq2Seq, RNN and LSTMs can be used to overcome the limitations of context recognition, which is important for response generation in NLP, allowing us to create better chatbots. The advances in deep learning can be leveraged and studied to build life-like chatbots which have natural conversations with humans.

For future research, as most chatbots generate text via a greedy approach, it would be integral to study text generation algorithms like beam search. To optimize computational performance, Gated Recurrent Units (GRUs) can be considered in place of existing LSTMs. To simplify and reduce the operations done by a network, GRUs combine the forget and input gates into a single update gate.

The responses by a chatbot are mostly just good guesswork, for computers to fully understand the complexity of human interaction, they would require a combination of multiple branches of science like psychology, literature and linguistics.

VII. HYPOTHESIS

Deep learning can allow us to create a chatbot model that incorporates the context preserving capabilities of LSTM cells along with the methodical approach of sequence to sequence architecture. Generative models seem to be the way to build a human-like chatbot, but the disadvantage lies in the grammatical errors that the model generates. They may also provide responses in inappropriate language, depending on what training data is used. This impedes the amount and quantity of data to be used to train a chatbot in a production level product. The chatbot model will be evaluated by metrics involving computer calculated scores and evaluations by people. The chatbot, acting as an actual person, will interact with people, and we will examine if a person can tell that it's a chatbot. The deep learning model will be evaluated against an existing retrieval-based model to give a clear idea of the significance of generative models and what can be improved. While giving appropriate replies to user is of utmost importance, another way to evaluate the chatbot is to observe if the context of the conversation is maintained. The chatbot model will also be assessed to see how well it can retain context in the ongoing conversation, as well, as examined after the context has been changed.

VIII. DATASETS

For training a chatbot model, the data need to be in an utterance-response pair form. The conversation should have a flow and context and should not be random utterances. Multiple sources given below have this kind of data:

1. Twitter Chat Log
2. Cornell Movie-Dialogs Corpus
3. Publicly available Reddit comments

The Movie corpus is vast and versatile, but the sayings are dramatic, and responses are often in questions. The Twitter log while extensive, contains incoherent spellings and phrases that are selectively used. The Reddit comments data, collected in 2015 and consisting of almost 1.7 billion comments, seemed to be the best training dataset. For this model, a random subset of 12 GB was selected to train the model, due to computation limitations [26].

Each comment in the dataset has metadata in JSON format containing ID and parent ID, etc. The comments are in levels of hierarchy of parent comment, child comment, sub-child comment and so forth. The hierarchy gives structure to create tuples in the form of pairs of comments and makes preprocessing a simple task. It can be understood below:

-Parent comment #1
--Child comment #1
---Sub-child comment #1
---Sub-child comment #2
--Child comment #2
-Parent comment #2
--Child comment #1

Here, pairs of comments will be formed like: -Parent comment #1 & --Child comment #1. The data format of each comment is in the form of an JSON object and its metadata is stored in fields like body, parent_id, author, edited, etc.

Example:

{"gilded":0,"author_flair_text":"Male","author_flair_css_class":"male","retrieved_on":1425124228,"ups":3,"subreddit_id":"t5_2s30g","edited":false,"controversiality":0,"parent_id":"t1_cnapn0k","subreddit":"AskMen","body":"I can't agree with passing the blame, but I'm glad to hear it's at least helping you with the anxiety. I went the other direction and started taking responsibility for everything. I had to realize that people make mistakes including myself and it's gonna be alright. I don't have to be shackled to my mistakes and I don't have to be afraid of making them. ","created_utc":"1420070668","downs":0,"score":3,"author":"TheDukeofEtown","archived":false,"distinguished":null,"id":"cnasd6x","score_hidden":false,"name":"t1_cnasd6x","link_id":"t3_2qyhmp"}

Out of the above fields, “parent_id”, “subreddit”, “body”, “score” and “id” will be used for training purposes.

After training on the Reddit data, it was observed that the chatbot responded with some inappropriate words at times and hence it was decided to train another model using the Movie Dialogs Corpus [27]. This is an extremely well-formatted dataset of dialogues from movies. It has 220,579 conversational exchanges between 10,292 pairs of movie characters, involving 9,035 characters from 617 movies with 304,713 total utterances. In both datasets, the inputs are fed to the model in form utterance-response pairs, so the model can learn the response occurrence based on user input.

IX. EVALUATION METRICS

A. BLEU Score:

The Bilingual Evaluation Understudy (BLEU) score, proposed by Papineni et al. is an algorithm that was originally designed to evaluate the quality of machine-translated natural language text [17]. It can be used to compare a generated sequence of words to a reference sequence. Although it was initially created for translated text, it can be generalized to evaluate most NLP problems. BLEU is a good metric because it is easily interpretable and computationally fast. It has frequently been shown to be an accurate model as compared to human judgement and is considered as a standard benchmarking tool for NLP models.

The algorithm works using n-grams to compare and assign sentences an appropriate score. The machine output is compared with the reference output to find out the n-grams or the number of times a word appears, then an average is taken over it. BLEU is calculated using the below formula:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

Where BP stands for brevity penalty. This brevity penalty exists so that a high-scoring machine output should match the reference response in length. The “reply” from the Reddit comment-reply pairs from the dataset will be used as the reference string. For every user input, the model will generate a new response. BLEU scores will be calculated for each response. After scoring 50 responses, they will be averaged and considered as an aggregated score. A major drawback of this metric is that the order of words is disregarded. The score more accurately reflects just the number of times the word appears in the machine generated sequence. This is why the

BLEU scores are only close to human evaluation but cannot match human evaluation. This was what led to using the Turing Test for evaluation along with BLEU Scores.

B. Turing Test

Turing Test is a test to evaluate a computer's intelligent behavior to see if is indistinguishable from an actual person [7]. The test works like this: A human evaluator will examine a text conversation between a human and computer or interact with said computer via textual natural language conversations. If the evaluator is unable to differentiate between the man and machine or if the evaluator cannot determine it is talking to a computer, the test is passed. The Turing test is being used for evaluation because even though BLEU scores are standard now, it cannot outperform a human in evaluating the quality of a response.

Using a version of the Turing Test, the chatbot's responses will be scored to determine how realistic its responses are. Twenty people will be chosen to carry out the decided version of the Turing Test and they will not know that they are conversing with a chatbot. After conversing with the chatbot for more than 15 utterances, they will be asked*: "On a scale of 1 to 10, how natural did the conversation seem?"*. The scores were then normalized to achieve an aggregated score. Amongst, the people selected for the test, half of them will be selected from an engineering background and the other half from non-technical backgrounds, making the test more robust and thorough.

This version of the Turing Test is further modified when the evaluators interact with the TV character chatbot. It was made sure, during selection, that the TV character chosen was popular and known by all evaluators so as to determine authenticity.

X. DATA PREPROCESSING – WORD EMBEDDING

Since computers are unable to understand the concept of words, a mechanism for representing text is used to process natural language input. For an NLP task in deep learning, we use a preprocessing layer in the RNN architecture called an Embedding Layer. The mechanism for text mapping is a called word vector, which involves representing words or sentences using vectors of real numbers. Neural word embeddings allow context to be captured in sentences by learning the distributional semantics of the words.

Each word in the training data set can be represented using a one-hot encoded vector. A vector is created for each word in the word corpus and the vector size is the word vocabulary size of corpus, in our case ~7000 words. In this way, each word is represented by a binary vector of size 7000. The whole vector is sparse i.e. filled with zeroes, except for the index of the integer, which is marked with a 1.

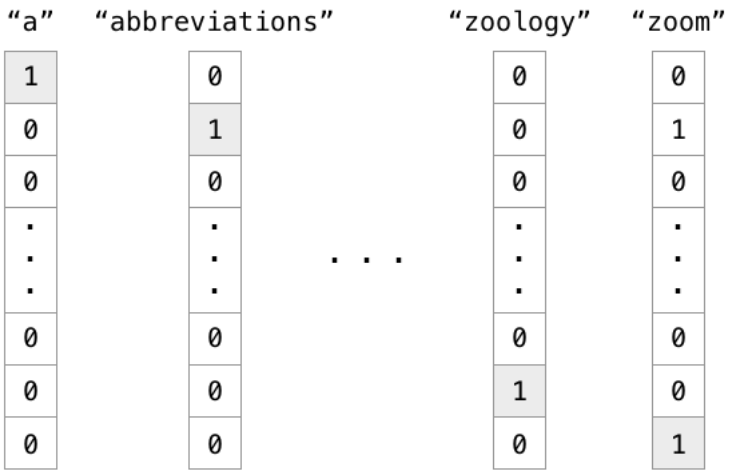


Fig. 8. One-hot encoding word vectors

One-hot encoded vectors are widely used as they are easy to implement. The drawbacks are that since there is no order, the context of words is lost, and neither is frequency occurrence

information considered. Simply creating one-hot encoding vectors for the words in our vocabulary is not enough, as the model cannot determine the relationship between words.

Word embeddings are used to capture context and represent the relationship between words. The most popular models for creating and learning word embeddings are Word2Vec and GloVe. Word2vec takes a large corpus of text as input and creates a vector dimensional space, usually of several hundred dimensions. Each word in the corpus is assigned a vector in the vector space and positioned so that words which have similar contexts in the corpus are in close proximity.

Word2vec can utilize either of two model architectures to produce a distributed representation of words: continuous bag-of-words (CBOW) or continuous skip-gram. In the continuous bag-of-words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (bag-of-words assumption). In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than more distant context words. According to the authors, CBOW is faster while skip-gram is slower but does a better job for infrequent words.

For preprocessing, a pre-trained Word2Vec model is used to create word embedding vectors for our corpus. The generated word vectors will then be used for training the chatbot encoder-decoder model.

XI. CONTEXT RETENTION

The challenge of our chatbot is to capture the context of the conversation, to make it more human-like. The problem can be perceived as paying more attention to certain words while generating output. For example, consider the sentence "I won a hotdog eating _____". Intuitively, we can predict the next word as "competition". When we are generating the word in blanks, we want to pay attention to the words "won" and "eating" [16].

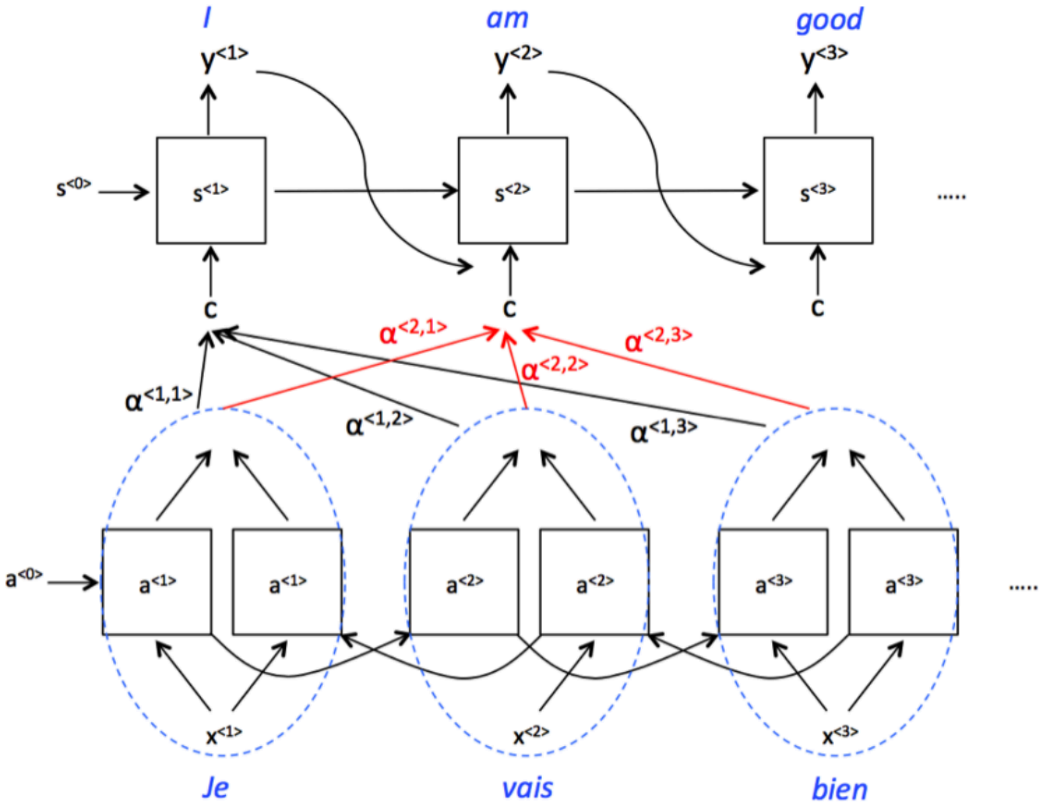


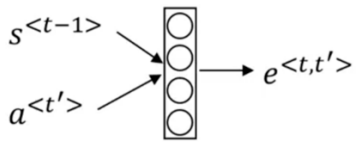
Fig. 9. Attention Capturing model proposed for Machine Translation [16]

Formally, we can represent this as a parameter α<t, t′> which represents the attention given to t′ token to generate t. It can be visualized in a network. α<1,2> represents the contribution of "vais" for generating the English translation of "Je".

The values for α can be calculated by the formula given below:

$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$

The parameter e can be learned using a small neural network as defined below:



The attention weights will transfer the attention from the encoder to the decoder for machine translation [16]. Since the output of the decoder is fed as input to the next time step, we consider s<t-1> and a<t′>, the output of the encoder at time t′ to calculate e<t, t′>. Since this relation is difficult to determine and can be different for each case, we will use gradient descent to learn the representation. In this way, we have included context in our model.

XII. MODEL PERFORMANCE AND RESULTS

Training Setup:

The model was built using Python tools including TensorFlow, Pandas, NumPy and Jupyter Notebook. The model was trained on a multi-GPU machine with two NVIDA Tesla 1080 GPUs, each 16 GB memory.

1. Unidirectional LSTM cells with Attention – Trained on Reddit Data

The model is a Seq2Seq architecture with Unidirectional LSTM cells with an attention mechanism at the decoder. We use a Beam Search algorithm in the LSTM cells to search for the next token or word in any given sequence of words i.e. a sentence. Rather than greedily choosing the most likely next word as the sequence is constructed, the beam search expands all possible next words and keeps the *k* most likely, where *k*, a user-specified input, decides the number of beams or searches through the sequence of probabilities.

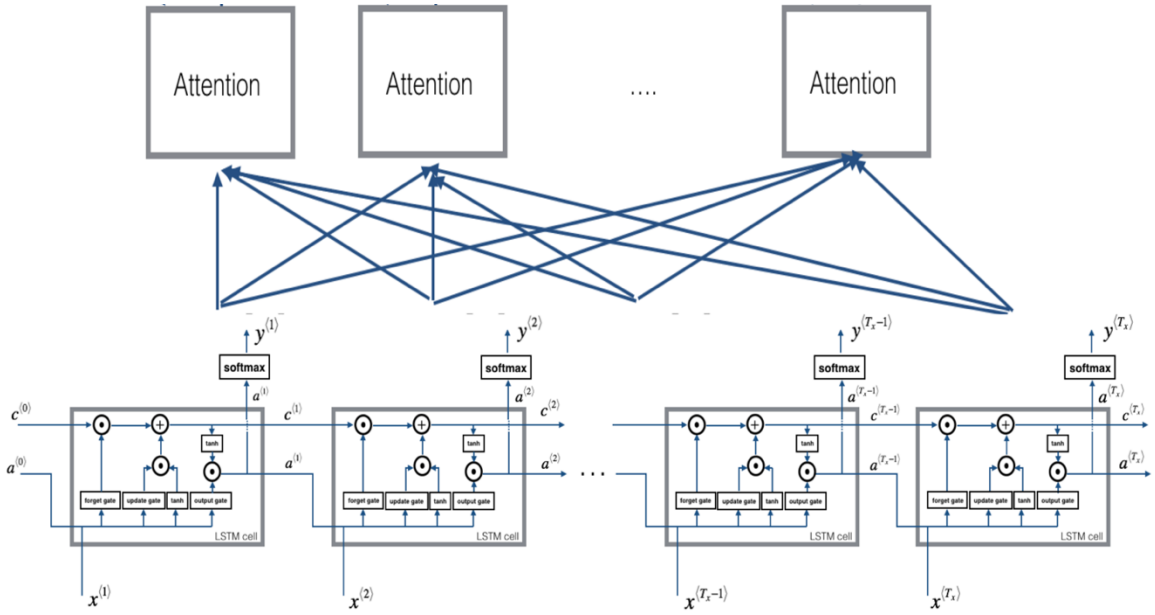


Fig. 10. Encoder-Decoder with Attention Mechanism

a<0> and c<0> represents the hidden state and state of memory cell at time step 0, respectively. x<1> is the input at the first timestep, which is the first input word in our case. In LSTMs, the hidden state is propagated to the next time step, which helps it to retain context. In case of LSTM, the more timesteps we look at significantly adds to the complexity of the model, ultimately increasing the runtime for training.

To indicate the end of a sentence, we use a <eos> tag to the end of the input sequence. This way the model understands when the sentence is over. Longer sentences often end up being complex, because the probability of every next word is embedded more as the sequence gets longer. Thus, for the sake of simplicity, the model only consumes 15 words at a time, and generates 15 words for the response.

A. Sample Conversation:

Here the user is denoted by “Human” and the chatbot is denoted by “Chatbot”

Human: Hello

Chatbot: Thanks for the clarification, I apologize for misunderstanding your comment.

Human: No I dont want to talk to you

Chatbot: That's what I'm saying. You don't know what you're talking about, you're talking to yourself.

Human: Will we ever get along?

Chatbot: What do you think I am talking about?

Human: Football

Chatbot: Are you sure you're not talking about football? I don't understand what you're saying.

Human: What do you think about cigarettes

Chatbot: That's not how things work. That's just what you're saying. That's what you're saying.

Human: Oh am I?

Chatbot: What are you talking about?

Human: I meant that this car is blue

Chatbot: You're talking about cigarettes, not cigarettes. That's not what cigarettes are for. That's cigarettes.

Human: I meant the car looks blue

Chatbot: I don't think you understand the difference between blue and yellow.

Human: What do you think about cigarettes?

Chatbot: I don't think you understand the difference between cigarettes and cigarettes.

B. Results and Analysis:

BLEU Score: 0.73

Modified Turing Test Score: 0.61

Training time: 6 hours 35 minutes

Loss: 0.3045

The chatbot appears to be speaking coherent English but seems to repeat itself often. The responses are often aggressive and defensive which accurately reflects the tone of the Reddit data where comment threads often consist of users arguing with each other. As noticed by the model's responses to user utterances on "football", "cigarettes" and the color "blue", it can pick up on context using its attention mechanism. It also seems to recall previous inputs as observed in response to User's input: "*I meant that this car is blue*", where the model generates an output containing a reference to cigarettes, which the user asked about previously. The final BLEU score is a good indicator of the model's performance as the replies are coherent, albeit random at times.

The modified version of the Turing Test reveals more, as most of the evaluators could tell it was a bot. However, the majority of evaluators were surprised by how coherent the responses were as well as the snarky personality that came from the Reddit data.

2. Unidirectional LSTM with Attention – Trained on Movie Dialog Corpus

The model has the same Seq2Seq architecture with Unidirectional LSTM cells with an attention mechanism at the decoder. For this model, instead of Beam Search, a greedy approach is used which means it takes the most likely word or token at each time step. This model was trained on the Movie Dialog Corpus to see how a model trained on a different dataset and heuristic algorithm would respond to user utterances [27]. Instead of 15, the sequence length is limited to a maximum of 5 to keep the utterances short to see how coherent the responses are to user utterances. It will also allow a comparison of the previous model with the current in terms of the modified Turing Test i.e. how human the responses seem to the evaluator.

A. Sample Conversation:

Here the user is denoted by “Human” and the chatbot is denoted by “Chatbot”

Human: Hi
Chatbot: Hi
Human: How're you?
Chatbot: I'm fine.
Human: Where're you from?
Chatbot: California.
Human: Which city?
Chatbot: Let's talk about something else.
Human: You don't like california?

Chatbot: Oh...

Human: Why not?

Chatbot: I can't.

Human: How old are you?

Chatbot: Eighteen.

Human: What do you do?

Chatbot: I'll be there.

Human: Where do you work?

Chatbot: In the trunk.

Human: Are you a mechanic?

Chatbot: I'm not.

Human: What are you?

Chatbot: No idea.

B. Results and Analysis:

BLEU Score: 0.79

Modified Turing Test Score: 0.73

Training time: 4 hours 15 minutes

Loss: 0.2933

The BLEU score of 0.79 indicates the model's language ability is high, given its grammar and syntax abilities. While this could be attributed to the short length, it is actually a combination of the short length and the greedy heuristic approach. At first glance the chatbot seemed very human to the evaluators as reflected in the modified Turing Test score.

The attention mechanism is working pretty well as it replies to each user input which appropriate answers. When asked about work, age and location, the model responded with fitting responses that accurately answered the questions, something the previous model could not do.

This credit can be given to the training data, as movie dialogs are more synchronous to utterance-response pairs than Reddit comments. The model also reflects the nature of the movie dialog dataset by giving dramatic answers such as "*Let's talk about something else*", "*I can't*" and "*No idea*".

3. Unidirectional LSTM with Attention – With Decoder "TV Character" Injection

Using the model trained in Part B. of this section, another model was created by training the decoder on a dataset such that the occurrences of the model would reflect the dataset the decoder has been re-trained on. For this purpose, the FRIENDS transcripts dataset was chosen [28]. In this dataset, the character 'Joey Tribbiani' has the most lines and hence his conversations with other characters were used as utterance-response pairs for training. This model too looks only looking for 5-word sentences, and thus this data is suitable as Joey lines are often short liners. Once again, a greedy heuristic for word selection is used during sentence generation.

A. Sample Conversation:

Here the user is denoted by "Human" and the Joey chatbot is denoted by "JoeyBot"

Conversation 1:

Human: Hi

JoeyBot: Hey.

Human: Who are you?

JoeyBot: Joey Tribbiani!

Human: Do you like food?

JoeyBot: Yeah.

Human: Rachel likes Ross?

JoeyBot: That's right.

Human: Do you like Phoebe?

JoeyBot: Oh yeah.

Human: Bye

JoeyBot: Bye.

Conversation 2:

Human: Sup buddy

JoeyBot: Je m'appelle Claude.

Human: You are so funny

JoeyBot: You are!

Human: What's up?

JoeyBot: Bad news.

Human: What happened to you?

JoeyBot: I know, but I don't know.

Human: I love you Joey

JoeyBot: I love you too.

B. Results and Analysis:

BLEU Score: 0.76

Modified Turing Test Score: 0.79

Training time: 5 hours 23 minutes

Loss: 0.4601

The model does a good job of mimicking the character Joey and picking up the mannerisms of the character as evident from the sample conversations. Particular character traits like being flirtatious or dumb-witted were exhibited in certain lines, like when asked about Phoebe, saying "*I love you too*" and answering "*I don't know*" often. This analysis was shared by the evaluators as well, with the model's score being an average of 8 on the modified Turing Test. For this model, the evaluators were asked, *"How much does the bot sound like Joey from a scale of 1 to 10?"*.

The BLEU score of 0.76 indicates the model's language capability is high, again attributed to a combination of the short length and the greedy heuristic approach. Appropriate responses are given to user utterances enough to hold a small conversation. The attention mechanism works fine as displayed when asked about its name, it gives the character name, thus picking up on context.

CONCLUSION

All three chatbot models achieve high scores in the BLEU evaluation which determines the language ability as well as appropriate response capability. The models pick up on context well as they have attention mechanisms and are thus able to give appropriate replies to user utterances, thus justifying the BLEU score. This factor also comes into play when analyzing the modified Turing Test score which is also decent for all models except Model-1 which is significantly lower at 0.61.

All three models effectively pick up on the tones and mannerisms reflected in the conversations in the dataset, that constitute of a personality. For e.g.: Model-1 responds in a snarky fashion, while Model-2 gives short but dramatic answers, as one would expect from dialogs in movies. Model-3 i.e. JoeyBot brings to life the traits of Joey Tribbiani by displaying a fun, enthusiastic nature in its responses.

The models trained on the movie corpus with shorter sequence lengths have higher scores in both evaluations. Model 2 i.e. the Reddit model gave incomprehensive answers at times due its larger sequence length which resulted in some complex sentences. Thus, we can conclude that training models on more conversation-based data seems to be more efficient while also limiting output sequences to smaller lengths.

XIII. FUTURE WORK

The chatbots in this project are created using unidirectional LSTM units, which mean they preserve information of the past because the only inputs it has seen are from the past. In the context of NLP and LSTMs, past and future refer to the previous and next words at any word in a sentence. Using bidirectional LSTM units, the cell will run over your inputs in two ways, one from past to future and one from future to past. What differs this approach from unidirectional is that, in the LSTM that runs backwards you preserve information from the future and using the two hidden states combined you are able to, in any point in time, preserve information from both past and future. Using the information from the future could be easier for the network to understand what the next word is.

Another method that could be used for training chatbots is Deep Reinforcement Learning. In this an agent interacts with an entity and the agent is rewarded when it produces desired results. This “reinforces” positive behavior in the agent. Such an algorithm would allow chatbots to be rewarded for a complete conversation rather than just individual sentences.

REFERENCES

[1] M. Jadeja, N. Varia and A. Shah, "Deep Reinforcement Learning for Conversational AI," *SCAI'17-Search-Oriented Conversational AI*, *San Diego,* 2017.

[2] I. Sutskever, O. Vinyals and Q. Le, "Sequence to Sequence Learning with Neural Networks," *NIPS '14 Proc. of the 27th Int. Conf. on Neural Inform. Processing Syst. - Volume 2, Montreal, Canada,* 2014.

[3] O. Vinyals, Q. Le, "A Neural Conversational Model," *ICML Deep Learning Workshop*, *Lille, France,* 2015

[4] P. Goyal, S. Pandey and K. Jain, "Developing a Chatbot," *Deep Learning for Natural Language Processing, Apress, Berkeley, CA,* 2018. pp 169-229.

[5] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao and D. Jurafsky, "Deep Reinforcement Learning for Dialogue Generation," *Proc. of the 2016 Conf. on Empirical Methods in Natural Language Processing, Austin, Texas,* 2016.

[6] Xu, Z. Liu, Y. Guo, V. Sinha and R. Akkiraju, "A New Chatbot for Customer Service on Social Media," *ACM, New York,* 2017

[7] A. Turing, "Computing Machinery and Intelligence," *Mind*, *vol. 236, October 1950*, pp. 433-460.

[8] I. Thies, N. Menon, S. Magapu, M. Subramony and J.O'Neill, "How Do You Want Your Chatbot? An Exploratory Wizard-of-Oz Study with Young, Urban Indians," *Human-Comput. Interaction - INTERACT 2017, LNCS, vol 10513. Springer, Cham, Mumbai, India*, 2017. pp 441-459.

[9] M. Qiu, F. Li, S. Wang, X. Gao, Y. Chen, W. Zhao, H. Chen, J. Huang and W. Chu "AliMe Chat: A Sequence to Sequence and Rerank based Chatbot Engine," *Proc. of the*

55th Annu. Meeting of the Association for Computational Linguistics, Vancouver, Canada, 2017. pp 498–503.

[10] S. Mishra, D. Bharti, N. Mishra and "Dr. Vdoc: A medical Chatbot that acts as a Virtual Doctor," *Research & Reviews: Journal of Medical Sci. and Technol., Vol 6, No 3*, 2017. pp 16–20.

[11] J. Abrams, "Is ELIZA human, and can she write a sonnet? A look at language technology," *ACCESS, Volume 31, Issue 3,* 2017. pp 4-11.

[12] "PARRY", [Online Available]: https://en.wikipedia.org/wiki/PARRY

[13] R. Wallace, "Anatomy_of_ALICE," *Parsing the Turing Test, Springer, Dordrecht*, 2009

[14] [Online Available]: http://www.jabberwacky.com/j2about

[15] [Online Available]: https://en.wikipedia.org/wiki/SmarterChild

[16] Bhagwat, Vyas Ajay, "Deep Learning for Chatbots" (2018). *Master's Projects. 630.* [Online Available]: https://scholarworks.sjsu.edu/etd_projects/630

[17] K. Papineni, S. Roukos, T. Ward and W. J. Zhu, "BLEU: a method for automatic evaluation of machine translation," *ACL-2002: 40th Annual meeting of the Association for Computational Linguistics., p. 311–318, 2002.*

[18] C. Lee, Y. Wang, T. Hsu, K. Chen, H. Lee and L. Lee. "Scalable Sentiment for Sequence-to-Sequence Chatbot Response with Performance Analysis," *2018 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* 2018. pp 6164-6168.

[19] W. Yin, K. Kann, M. Yu and H. Schütze, "Comparative Study of CNN and RNN for Natural Language Processing," 2017.

[20] R. Collobert, J. Weston, “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning,” *Proc. of the 25th Int. Conf. on Mach. Learning, Helsinki, Finland,* 2008.

[21] H. Li, “Deep learning for natural language processing: advantages and challenges,” *Nat. Sci. Review, Volume 5, Issue 1*, 1 January 2018. pp 24–26.

[22] O. Davydova, “7 types of Artificial Neural Networks for Natural Language Processing,” [Online Available]: https://medium.com/@datamonsters/artificial-neural-networks-for-natural-language-processing-part-1-64ca9ebfa3b2

[23] Y. LeCun, Y. Bengioz, G. Hinton, “Deep Learning,” *Nature, Vol. 521,* 28 May 2015. pp 436 – 444.

[24] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Networks, Vol 61,* Jan 2015. pp 85-117.

[25] “FRIENDS Transcripts”, [Online Available]: https://fangj.github.io/friends/

[26] “Reddit Comments Dataset”, [Online Available]: https://www.reddit.com/r/datasets/comments/65o7py/updated_reddit_comment_dataset_as_torrents/

[27] “Cornell Movie Dialog Corpus”, [Online Available]: https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html