SELF LEARNING AGENTS FOR CHATBOT APPLICATIONS

A DISSERTATION

Submitted in partial fulfilment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY
in
ADVANCE COMPUTING AND DATA SCIENCE

By
PARVATHI SAXENA
179305003

MANIPAL UNIVERSITY
JAIPUR

(In collaboration with **C-DAC**)

Computer Science & Enineering
School of Computing & Information Technology

MANIPAL UNIVERSITY JAIPUR
JAIPUR-303007
RAJASTHAN, INDIA

May-2019

SELF LEARNING AGENTS FOR CHATBOT APPLICATIONS

A DISSERTATION

Submitted in partial fulfilment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY
in
ADVANCE COMPUTING AND DATA SCIENCE

By
PARVATHI SAXENA
179305003

MANIPAL UNIVERSITY
JAIPUR

(In collaboration with **C-DAC**)

Computer Science & Enineering
School of Computing & Information Technology

MANIPAL UNIVERSITY JAIPUR
JAIPUR-303007
RAJASTHAN, INDIA

May-2019

MANIPAL UNIVERSITY
JAIPUR

MANIPAL UNIVERSITY JAIPUR

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the dissertation entitled "SELF LEANING AGENTS FOR CHATBOT APPLICATIONS" in partial fulfilment of the requirements for the award of the Degree of Master of Advanced Computing and Data Science submitted to the Department of Computer Science and Engineering, School of Computing and Information Technology, Manipal University Jaipur, Jaipur is an authentic record of my own work carried out during the period from July 18 to April 19 under the supervision of Dr. Akhilesh Kumar Sharma of Department of Computer Science and Engineering.

The matter presented in this dissertation has not been submutted by me for the award of any degree of this or any other institute.

Date: Signature

Place: Parvathi Saxena

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: Signature

Place: Dr. Akhilesh Kumar Sharma

CERTIFICATE

This is to certify that the dissertation report entitled

" SELF LEARNING AGENT FOR CHATBOT APPLICATIONS"

Submitted by

Parvathi Saxena

is a bonafide work carried out by her without the Sponsorship from C-DAC

under the

supervision of Mr. Samrit Kumar Maity and has been completed successfully.

Mr. Samrit Kumar Maity

Place :

Date :

ACKNOWLEDGEMENT

I owe my gratitude and appreciation towards **Mr. Samrit Kumar Maity, Mr. Manish Kumar Nirmal** and **Dr. Akhilesh Kumar Sharma** for their kind co-operation and encouragement which helped me in the completion of this seminar.

I would also like to thank all the staff members of **MANIPAL UNIVERSITY** and **C-DAC** for their valuable assistance and support during my academia.

Finally, yet importantly, I would like to express my heartfelt thanks to my beloved parents for their blessings, my friends/classmates for their help and wishes for the successful completion of this report.

Place:

Date:

- Parvathi Saxena

ABSTRACT

With the emerging trend of chat applications across the globe has made chatbots a new technology that has changed the way we do our conversations. It is a technology that brings many advantages with it for businesses that adopt it. Having an interactive chatbot whose narrative is not programmer driven but instead its own would give rise to a even stronger virtual companion. But these chatbot applications have certain shortcomings. This report aims to highlight those facts and has proposed a model that can learn better as compared to other popular approaches. The proposed model has be trained on various bAbI tasks and compared its performance against LSTM based model and Dynamic Memory models. We are able to successfully conclude that GRU outperformed 50% of the tasks.

Keywords: LSTM; GRU; RNN; response generation; deep learning; machine learning; Artificial Intelligence Markup Language(AIML).

CONTENTS

Acknowledgement		i
Abstract		ii
Contents		iii
List of Figures		iv
List of Tables		v
Abbreviation		vi
1	Introduction	1
1.1	Types of Chatbots	3
1.2	Classical Approach	4
2	Motivation	5
3	History	7
4	Popular Neural Network Models	9
4.1	Recurrent Neural Network (RNN)	9
4.2	Long Short-Term Memory (LSTM)	10
4.3	Gated Recurrent Unit (GRU)	11
4.4	Vanishing/ Exploding Gradients	13
6	Literature Survey	15
7	Technical Specifications	22
7.1	Environmental Setup	22
7.2	Python Packages	22
7.3	bAbI Dataset	22
8	Proposed Work	28
8.1	Results	29
9	Conclusion	31
10	Project Work Schedule	32
10.1	Overall Project Status: On Schedule	32
11	Paper Publication	33
12	References	39

LIST OF FIGURES

Figure No	Figure Description	Page No.
Figure 1	Types of Chatbots	4
Figure 2	LSTM	11
Figure 3	GRU	12
Figure 4	Recurrent Neural Networks (RNN)	15
Figure 5	Sequence-to-sequence	16
Figure 6	Working of XML-CNN	17
Figure 7	Question Module	19
Figure 8	Block Diagram	29
Figure 9	Output for Agent's Motivation Task	31
Figure 10	Bar Chart on Accuracy of Models	32

LIST OF TABLES

Table No	Table Description	Page No.
Table 1	Accuracy Comparison between LSTM, GRU and Dynamic Memory	30
Table 2	Time Line Chart for Dissertation	41

ABBREVIATIONS

Acronym	Abbreviation
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
ML	Machine Learning
AWS	Amazon Web Services
LUIS	Language Understanding Intelligent Service
XMPP	eXtensible Messaging and Presence Protocol
LSA	Latent Semantic Analysis
AIML	Artificial Intelligence Markup Language
FAQ	Frequently Asked Questions
AI	Artificial Intelligence
GRU	Gated Recurrent Unit
RNN	Recurrent Neural Network

CHAPTER 1

INTRODUCTION

AI in marketing is booming now in all industries like medical, manufacturing etc. These large industries are moving towards the use of chatbots as they can get an insight into the user's behavior. This approach of integrating certain industry services to chatbots has completely changed the marketing approaches. Now companies can interpret exactly what their customers are like and what they do on the internet. Emailing has been frowned upon as inboxes get filled with them and spam causing users to ignore them. So, experts have decided to move to social media as they have a wide access to data that can help in the expansion of their services and products which in turn allows them to engage more customers that leads to maximizing their sales. Most people today prefer to use only just five apps that are WhatsApp, Messenger, Twitter, Instagram and Viber. Most Messaging applications like WhatsApp, Messenger, Telegram etc. are having over 5 billion active users every day. For the first time in history people are using them more than social networks, i.e. it's fair to say that users are using most of their time on WhatsApp and messenger than they do on Twitter or Facebook. This showed that chatting is a more engaging activity. Brands need to engage with their customers through these messaging mediums and chatbots are the solution. Chatbots are a type of program that can converse with a user, they mimic both user's conversational ability and performance capability. They can communicate and perform basic tasks such as answering questions or placing product orders. Their main purpose would be to streamline interactions between people and services. Since the general public prefers messaging over having conversations over the telephone, customer support could also be provided through chatting streams. This would also stop the waiting period made by customers to be assigned to a customer support agent for telephonic support.

Chatbot is a conversational agent which conducts conversations via text or voice commands. They are gaining popularity as a crucial component of real-world applications since they can cater the customer experience by providing conversational marketing. They never sleep, they are cost efficient and they work well to work with all sorts of metrics. The idea of automating and scaling one-to-one conversations using this technology appeals to lots of brands and services out there. In this process, designers play an important role at defining how each conversation is scripted and the behaviors users can expect when interacting with bots. But, despite the best of our intentions, sometimes chatbots fail to deliver user experiences that are

as seamless, delightful and efficient as we envisioned them to be. Here are a few common reasons for chatbots to fail:

1) AI (Artificial Intelligence) is still not accessible

   The vast majority of chatbots aren't intelligent. They are built based on a decision-tree logic, where the response given by the bot depends on specific keywords identified in the user's input. E.g.

   Input: ***'shop' or 'buy***

   Output: message with product list

   What that means is that decision-tree types of bots are as intelligent as the capacity (and thoroughness, and patience) of the designer/programmer who created it to anticipate all potential user use cases and inputs. Bots with linguistic and natural language learning capabilities are still quite rare.

2) Use cases are not that strong

   Any new technology is out in the market it urges designers and developers to get excited to use them in all application scenarios they can. This kind of the approach doesn't give relevant use cases and in turn offers poor experiences. It's a natural part of the cycle, where industry needs to learn from their failures before it can deploy bots that are truly relevant and smart.

3) Lack of transparency

   The most successful bots out there make one thing clear from the very beginning of the experience: that the user is chatting with a robot, not with another human. Setting up the right expectations upfront will make users more forgiving about certain mistakes the bot might make. You certainly want your bot to feel as human as possible but lying to your users and pretending to be something it is not can lead to irreversible loss of trust.

4) They fail to understand the context

   Unlike humans chatbots do not really understand sarcasm or read between the lines. It cannot keep up with a conversation as it fails to leverage contextual information to a particular response. Except in cases where bots are powered by natural language processing technology, they can't hold contextual information for longer than a few chat

bubbles and will end up losing track of what the user was saying before they posed the question.

5) They don't communicate with existing business systems
When re-creating its functionality from scratch can be harmful for both businesses and customers. Designers and developers tend to get excited about all the tasks a bot can help with but forget to narrow its area of focus. They try to address problems that are beyond their scope.

6) Lack proper human escalation protocols
When technology fails, users still want to be able to rely on human beings to help them solve their problems. Still, very few chatbots have an escalation workflow in place to let a human take over the conversation when the bot is unable to help.

1.1 Types of Chatbots

The idea is to create a chatbot that doesn't require human intervention in keeping a conversation going. This is where deep learning models come into picture. Using NLP along with different NN models allow this process. But these approaches are divided into Generative or Retrieval based models.

Generative dialogue systems, as their name indicates, they generate utterances word by word to a given conversation history (that we call context). Most of these systems are based on the sequence-to-sequence architecture in which the context is encoded into a summary vector. Then, this vector is decoded into a response or the next utterance. They suffer from generalization since they tend to generate predefined outputs like "thank you", "Ok" etc. most of the time. Their strength is related to their capacity to generate responses even if they do not understand the corpus.

Retrieval-based models have a repository of pre-defined responses they can use, which is unlike generative models that can generate responses they've never seen before. A bit more formally, the input to a retrieval-based model is a context $\boldsymbol{c}$ (the conversation up to this point) and a potential response $\boldsymbol{r}$. The model outputs a score for the response. To find a good response you would calculate the score for multiple responses and choose the one with the highest score. The problem is that generative models don't work well in practice. At least not yet. Because they have so much freedom in how they can respond, generative models tend to make grammatical mistakes and produce

irrelevant, generic or inconsistent responses. They also need huge amounts of training data and are hard to optimize. The vast majority of production systems today are retrieval-based, or a combination of retrieval-based and generative. Google's Smart Reply is a good example. Generative models are an active area of research, but we're not quite there yet.

The NLP module will create a structured data and collect anagrams like tf-idf and then encode them to one-hot encodings. This output is in a vector format which will be used for training the model. It is also a numerical kind of representation of the training set. We know we cannot provide any ML model with non-numeric data. So, word embeddings will convert them to vectors. The below diagram shows the different types of chatbot present.

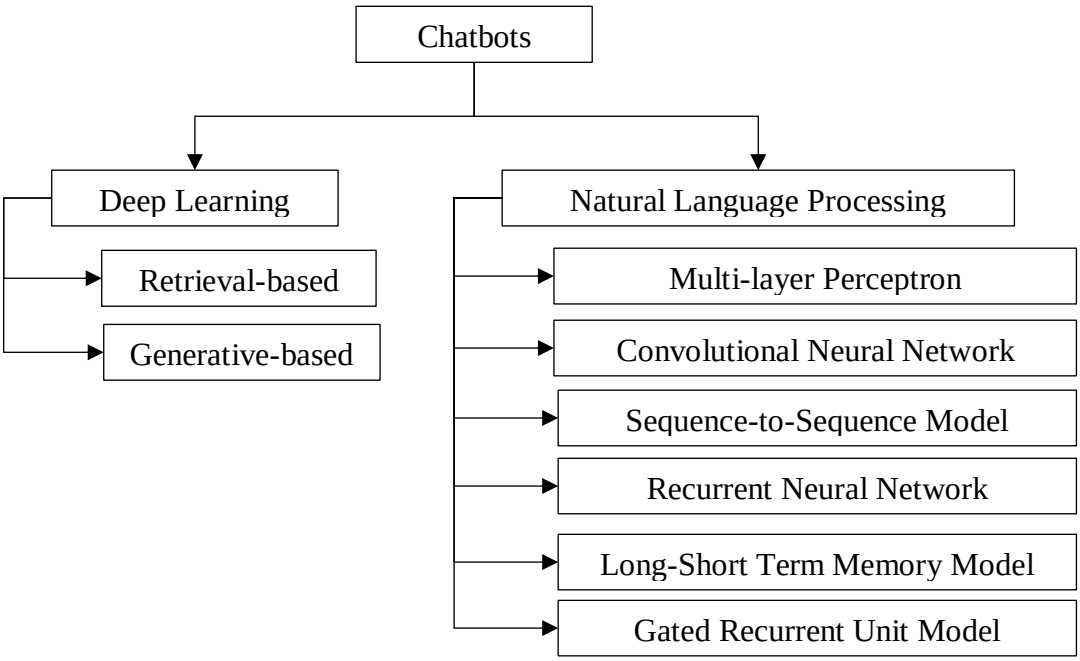

Fig.1: Types of Chatbots[13]

1.2 Classical Approach

Consulting groups like botwork.ai are using AI to create chatbots for brands like change.org, Tony Robins, Media Post etc. to help them engage in consumers across a variety of platforms tailoring the technology to each brand's specific needs. Major types of dialogue systems are,

1) Goal-Oriented
2) General Conversation
    a. Generative models
    b. Selective models

The idea is the same for both, the classical approach will be to (1) Build it, (2) Train it and (3) Test it. Knowledgebase (KB) should have a dialogue dataset since labelled datasets are easiest to learn from. We call each row in it i.e. a context-reply pair. The context could be one or several input sentences and the reply would be the label. Sometimes there is an EOS (End Of Sentence) token at the end of each sentence in the batch. This helps in the batch. This helps ML algorithms understand sentence bounds and the internal states.

CHAPTER 2

MOTIVATION

A primitive approach to building a chatbot would be to hard code every response for the chatbot. But this approach is limited to the capability of the programmer to generate all the possible responses any user could make[8]. With over 5 billion people among us and having one or a group of programmers create all possible combinations of responses is not feasible.

Generating a supervised model for getting trained on a set of input and when fed with similar data for testing showed some promising results but still did not work because the model is only as smart as its training data.

Conversations are ambiguous and unpredictable. So, researchers moved towards RNNs because it can handle variable length inputs[11]. But it was more challenging to train that model on just tokens from sentences fed to it. However, word embeddings were the way to go to capture links/relationships between the words. With the outputs generated by this technique was then fed to RNN model allowing the RNN model to learn.

The biggest challenge faced in the RNN models was vanishing/exploding gradients problem. Because of this problem the model was losing relevant information and keeping irrelevant information. This was when LSTMs[12] were introduced to meet these shortcomings. LSTM model was designed with a mechanism to remember relevant information and forget irrelevant information. But with the LSTM[13] model, if the information is too large and if the context of the conversation has never changed then remembering something that never left its memory was challenging to handle. So Dynamic Memory models were designed to keep such persistent information in its memory allowing the load on the LSTM[10] model to reduce and in turn generating smooth learning curve. But in the Dynamic Memory models the memory is divided into episodic and dynamic memory. Since LSTM model calculations are complex and it uses two different activation functions i.e. tanh and sigmoid. The outputs of both the functions are different it allows it to either keep information or loose it. But with episodic memory some relevant information got lost as the calculations were either to (0,1) or (-1,1) .

GRU was designed based on the LSTM model with only two gates and uses only one activation function. It merged the functionalities of the LSTM into two gates i.e. the forget and update gate. It was computationally less complex and worked well with dynamic memory models.

But the model still had shortcomings like understanding when the context changed in the conversation, so researchers introduced speaker ID feature to the dataset. This way the model

would keep track of who and what was spoken of. But more challenging was one unit cannot handle all these computations on its own. So using different models to split the computations of handling the conversations and their output fed to the final LSTM or GRU cell to get desired response.

From the literature review we are able to understand that sequential models can be trained on tokens fed to it from a sentence and that it can output many possible responses[9,12]. And we can conclude that using speaker IDs we can keep the track of the conversation context. So we can take an approach of using sequential models to handle the sentences and a GRU model to learn the conversations.

CHAPTER 3

HISTORY

Alan Turing first spoke of the concept of chatbots in the 1950s and since then the concept of chatbots has evolved through the decades. Alan Turing theorized that a truly intelligent machine would be indistinguishable from a human during a text-only conversation. These ideas essentially laid the foundations for the chatbot revolution.

In 1966, Eliza was created at the MIT AI laboratory to simulate human conversation by matching user prompts to scripted responses. It gave an illusion of understanding but had no built in framework for contextualizing events. It features the dialog between a human user and a computer program representing a mock Rogerian psychotherapist. ELIZA operates by recognizing key words or phrases from the input to reproduce a response using those keywords from pre – programmed responses. For instance, if a human says that 'My mother cooks good food'. ELIZA would pick up the word 'mother', and respond by asking an open- ended question 'Tell me more about your family'. This created an illusion of understanding and having an interaction with a real human being though the process was a mechanized one.

In 1972, while Eliza was a simulation of a therapist. Parry simulated a person with paranoid schizophrenia. It embodied a conversational strategy and was much more serious than Eliza. Parry was the first to pass the Turing Test - it was in the early seventies, when human interrogators, interacting with the program via remote keyboard, were unable with more than random accuracy to distinguish Parry from an actual paranoid individual.

Jabberwacky developed in the 1980s and released online in 1997, was a chatbot designed to simulate natural human chat in an interesting, entertaining and humorous manner. In 1992, an AI speech synthesis program created for MS-DOS based personal communications services assumed the role of a psychologist when interacting with others and was designed to showcase a digitized voice.

In 1995, the Artificial Linguistic internet Computer Entity(ALICE) was a NLP bot. She could apply heuristic pattern matching rules to human input in order to have a conversation, but was still not able to pass the Turing test. In 2001, an intelligent bot was widely distributed across SMS networks and Buddy lists of AOL and MSN messenger users. It offered a fun personalized conversation and was considered a precursor to apple's Siri and Samsung's Voice.

In 2006, IBM Watson was originally designed to compete on the TV show Jeopardy in which he beat two of the show's former champions. Watson has since gone on to bigger and better

things using NLP and ML to reveal insights from large amounts of data. In 2010, Siri is intelligent personal assistant. It is part of Apple's IOS and uses a natural language UI to answer questions and perform various requests. Siri did the groundwork for all later AI bots and personal assistants. Apple's first theory of a digital personal assistant was originally a concept in 1987, called the "Knowledge Navigator". Apple was the beginning head to set a voice-controlled assistant on a phone and Siri chatbot has grown from being a simple interface to an iPhone's limits to something considerably smarter, clever and faster.

In 2012, Google developed Google Now mobile app, it employs a natural language UI to answer questions, make recommendations and perform actions by passing on requests to a set of web services. In 2015, Alexa an intelligent personal assistant that inhabits the amazon Echo device, Alexa's capable of voice interaction, using NLP algorithms to receive, recognize, and respond to voice commands. In the same year Cortana, Microsoft's version of intelligent assistant that can set reminders and answer questions using the Bing search engine. Cortana recognizes natural voice commands and is available in a number of different languages.

In April 2016, Facebook launched a messenger platform which allows developers to create bots that can interact with Facebook users. At the end of 2016, 34,00 bots were available covering a wide range of use cases. Microsoft created Tay to mimic the speech and habits of a teenage girl. It caused controversy when it began to post offensive tweets and became increasingly paranoid. It eventually had to be shut down just 16 hours after launch.

CHAPTER 4

Popular Neural Network Models

4.1 Recurrent Neural Network (RNN)

In feed forward network, data just flows one way from the input node all the way to the output node. It only accepts data that is fixed size like an image or a number. Give it a labeled dataset like whether or not a temperature is hot or cold and it will be able to predict if a given temperature is hot or cold. But a conversation isn't a fixed size. It's a sequence of words. We need a network that can accept sequences as an input, i.e. a RNN. In a RNN, we feed the data back into input while training it in a recurring loop. RNNs are a type of ANN that add additional weights to the network to create cycles in the network graph in an effort to maintain an internal state. The promise of adding to NN is that they will be able to explicitly learn and exploit context in sequence prediction problems. RNNs used for Deep Learning are as follows,

1) Recurrent Neural Network
    a. Fully Recurrent Network
    b. Recursive Neural Network
    c. Neural History Compressor
2) Long Short-Term Memory
3) Gated Recurrent Unit
4) Neural Turing Machine

RNN are the state-of-the-art algorithm for sequential data and among others used by Apple Siri and Google Voice Search.

Two issues of standard RNNs are (1) exploding gradients and (2) vanishing gradients. RNN are a powerful and robust type of NN and belong to the most promising algorithm as they are the only ones with an internal memory. Because of their internal memory, RNNs are able to remember important things about the input they received which enables them to be very precise in predicting what's coming next.

Therefore they are preferred algorithm for sequential data like time series, speech, text, financial data, audio, video, weather and much more as they can form a much deeper understanding of sequence and its context compared to other algorithms. Sequential data is basically just ordered data where related things follow eachother. E.g. financial data, DNA sequence, time series data.

In RNN the information cycles through a loop. When it takes a decision it takes into consideration the current input and also what it has learned from the inputs it received previously. It has a short-term memory same as LSTM. It has two inputs, the present and the recent past. Because the sequence of data contains crucial information about what is coming next, which is why RNN can do things other algorithms cant. RNNs apply weights to the current and also to the previous input. They also tweak their weights for both through gradient descent and backpropagation through time.

RNNs can map to one to many, many to many and many to one. Back propagation is nothing but going backwards through your NN to find the partial derivatives of error ww.r.t to the weights which enables you to subtract this value from the weights. Those derivatives are then used by gradient descent an algorithm i.e. used to iteratively minimize a given function.

4.2 Long Short-Term Memory (LSTM)

In a LSTM model, it has the same control flow as RNN. It processes data sequentially, passing on information as it propagates forward. The difference are the operation within the LSTM cells. These operation are used to allow the LSTM to forget or keep information. The cell gate acts as a transport highway that transfers relative information all the way down to the sequence chain (memory of the network). It can carry information throughout the sequence processing and in theory even information from earlier time steps could be carried all the way to the last time step and thus reducing the effects of short-term memory. As it goes on its journey, it gets information added or removed to the cell state via gates. The gates are just different NNs that decides which information is allowed on the cell state. The gates learn what information is relevant to keep or forget during training.

Gates contain sigmoid activation. A sigmoid activation is similar to the tanh. Instead of reducing values to between -1 to 1, it squishes values between 0 and 1 i.e. helpful to update or forget data because any number getting multiplied by 0 becomes 0 causing values to disappear or be forgotten. The network can learn what data should be forgotten or what is important to keep.

The forget gate decides what information should be thrown or kept away. Information from a previous hidden state and information from the current input is passed through the sigmoid function values come out between 0 and 1. To update the cell state we have the input gate, first we passed the previous hidden state and the current input to a sigmoid function that decides which values will be updated by transforming values to be between 0 and 1. You also

pass the hidden state and current input into the tanh function to reduce values -1 and 1 to regulate the network.

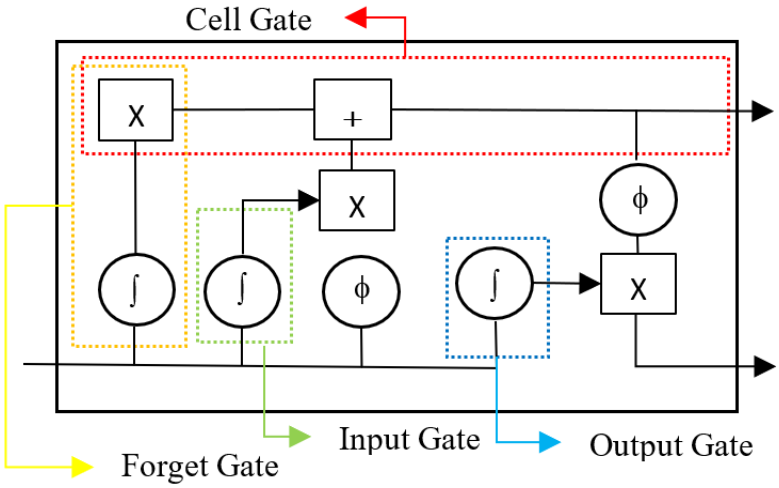

Fig.2: LSTM

Then you multiply the tanh output with sigmoid output, the sigmoid output will decide which information is important to keep from the tanh output. Now we get enough information to calculate the cell state. First the cell state is multiplied by the forget vector. This has the possibility of dropping values in a cell state if it is dropping values in a cell state if it is multiplied by values near zero.

Then we take the output from the input gate and do a polarize addition which updates the cell state to new values. This gives us our new cell state. The output gate, it decides what the next hidden state should be (the hidden state contains information of previous inputs).

The hidden state is also used for prediction. First we pass a previous hidden state an the current input until a sigmoid function. Then we pass newly modified cell state to the tanh function. We multiply the tanh output with sigmoid output to decide what information the hidden state should carry. The output hidden state, the new cell state and a new hidden state is then carried over to the next time step.

4.3 Gated Recurrent Unit (GRU)

A newer generation of RNN, similar to LSTM is the GRU model. It got rid of the cell state and used a hidden state to transfer information instead. It also has two gates, a reset gate and an update gate. The update gate acts similar to the forget and input gate of LSTM. It decides what information to keep or throw away. The reset gate is used to decide how much information to pass or to forget. They have less tensor operation therefore they are little faster to train than LSTM.

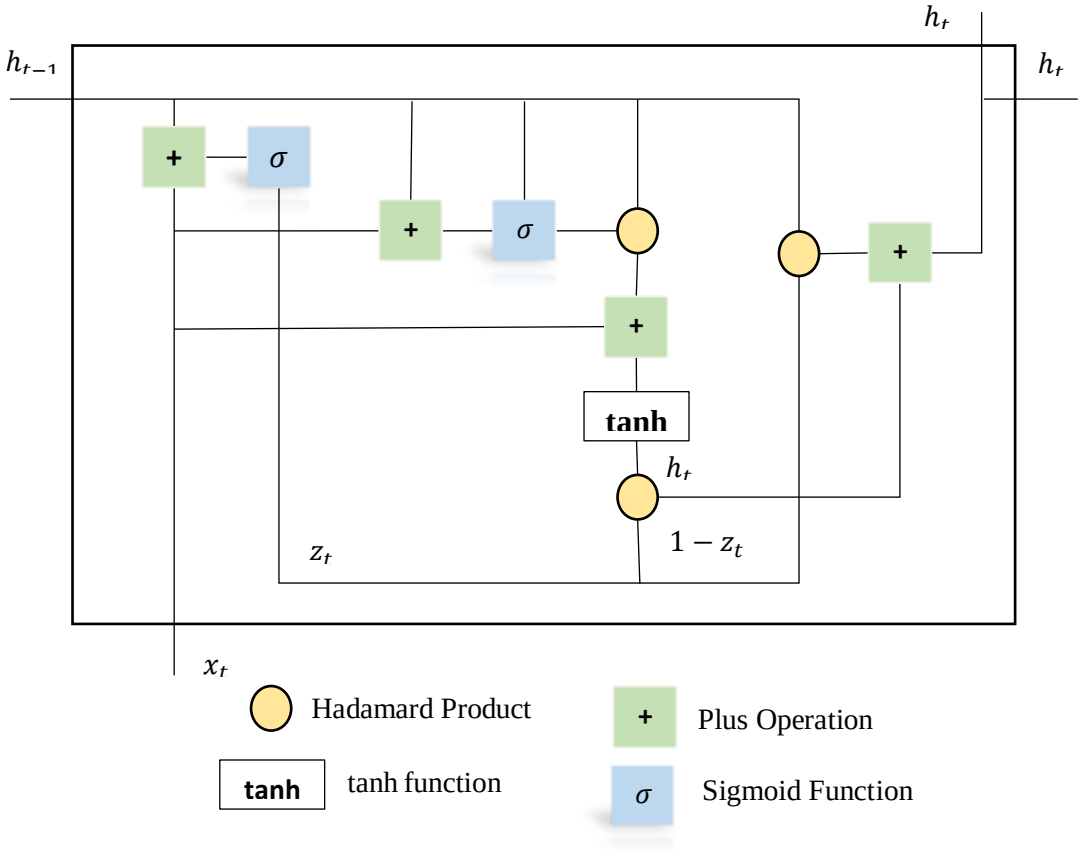

Fig.3: GRU

RNNs are good for processing sequence data for predictions but suffer from short-term memory. LSTMs and GRUs were created as a method to mitigate short-term memory using mechanism called gates. Gates are just NNs that regulate the flow of information being passed from one time step to next. LSTMs and GRUs are used in the state-of-art deep learning applications like speech recognition, speech synthesis, NLU etc.

GRU aims to solve the vanishing gradient problem which comes with a standard recurrent neural network. GRU can also be considered as a variation on the LSTM because both are designed similarly and, in some cases, produce equally excellent results. To solve the vanishing gradient problem of a standard RNN, GRU uses, update gate and reset gate. Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction.

In the update gate, first we calculate the $\boldsymbol{z_t}$ for timestep $\boldsymbol{t}$,

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

Here,

x_t = Word vectors

t = current timestep

h_{t-1}= Previous timestep vector

W = weights associated with the word vectors

U = weights associated with the previous timestep vector

σ = sigmoid function

z_t = update gate

When input is given to the unit i.e. x_t it is multiplied by its own weight W(z). Same happens for h_{t-1} for previous state. Both results are added together fed to a sigmoid activation function to get results between 0 and 1. The update gate helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future.

The Reset gate is used to decide how much of the past information to forget.

$$r_i = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right)$$

Here,

x_t = Word vectors

t = current timestep

h_{t-1}= Previous timestep vector

W = weights associated with the word vectors

U = weights associated with the previous timestep vector

σ = sigmoid function

r_i = reset gate

Similar to the update gate, only the weights and usage is different. The reset will learn to ignore the past timesteps when necessary. A new memory content is introduced which will use the reset gate to store the relevant information from the past.

$$h_t = \tanh(Wx_t + r_t \cdot Uh_{t-1})$$

Here,

x_t = Word vectors

t = current timestep

h_{t-1}= Previous timestep vector

W = weights associated with the word vectors

U = weights associated with the previous timestep vector

σ = sigmoid function

h_t = current timestep output

4.4 Vanishing/ Exploding Gradients

Problems faced during the training of a deep neural network is when the gradients are being propagated back in time all the way to the initial layer. The gradients coming from the deeper layers have to go through continuous matrix multiplication because of the chain rule and as they approach the earlier layers, if they have small values(<1), they shrink exponentially until they vanish and make it impossible for the model to learn, this is the vanishing gradient problem. While on the other hand if they have large values (>1) they get larger and eventually blow up and crash the model, this is the exploding problem.

When gradients explode, the gradients could become Nan because of the numerical overflow, or we might see irregular oscillations in training cost when we plot the learning curve. A solution to fix this is to apply gradient clipping, which places pre-defined threshold on the gradients to prevent it from getting too large and by doing this it doesn't change the direction of the gradients it only changes its length.

if $\|g\| > \text{threshold}$

$g \longleftarrow (\text{threshold X } g) / \|g\|$

Where,

g = gradient

$\|g\|$ = norm of the gradient

Another solution includes, redesign the network model i.e. to have fewer layers. In RNN, updating across fewer prior time steps during training called truncated back-propagation through time. It may reduce the exploding gradient problem. Another technique is weight regularization to check the size of network and apply a penalty to the network cost function for large weight values. Often an L1(Absolute weight) or an L2(Squared Weight) penalty can be used.

For dealing with Vanishing Gradients a simple solution is to identity RNN architecture, where the network weights are initialized to the identity matrix and the activation function are all set to ReLU. This encourages the network computation to stay close to the identity function. This works well because when the error derivatives are being

propagated backwards through time, they remain constants of either 0 & 1. Hence aren't likely to suffer from vanishing gradients. Another solution is LSTM, which can easily capture long-term dependencies in sequence data. The standard RNN operates in such a way that the hidden state activation are influenced by the other local activations closest to them, which corresponds to a short-term memory while the network weights are influenced by the computation that take place over entire long sequences, which corresponds to a long-term memory. Hence the RNN was redesigned so that it has an activation state that can also act like weights and preserve information over long distances.

CHAPTER 5

LITERATURE SURVEY

PAPER I: A Neural Conversational Model[8]

Starting with generative problems, they used a model called Sequence-to-sequence, to model the dialogue given to them in a dataset. This model is represented by 2 RNNs with a different set of parameters each. While regular feed-forward NN are given a new datapoint at every time step during training to learn from. RNNs are given both data points and the learned hidden state from the previous time step at each new time step during training. This way there is a recurrence in what its learning. Its learning from how its learned before a sort of recurrence or feed-back loop. This is useful for learning from a sequence of data.

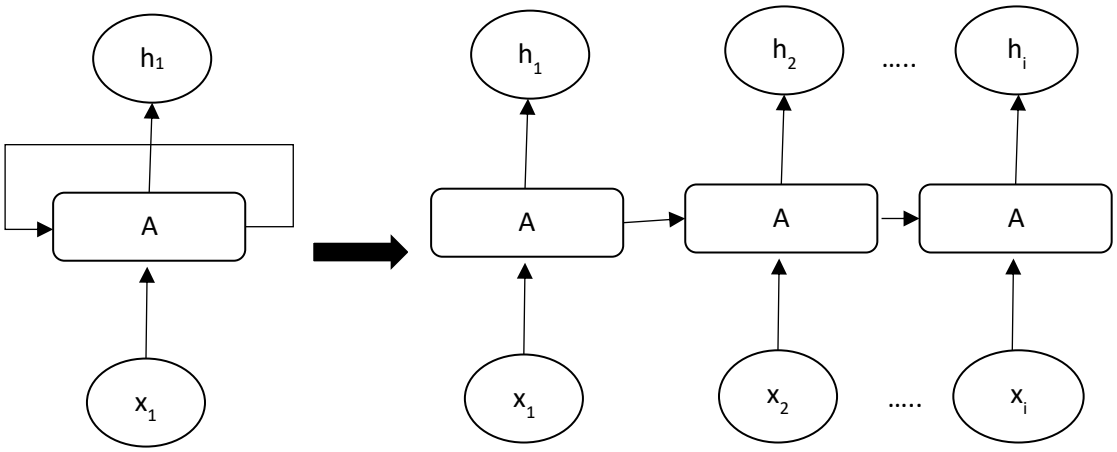

Fig.4: Recurrent Neural Networks(RNNs)

The first recurrent neuron is called the encoder, its given a sequence of context tokens one at a time and updates its hidden state accordingly after processing the whole context sequence. It produces a hidden state which incorporates a context and uses it for generating an answer. The other recurrent neuron is called the decoder, its job is to take the context representation from the encoder as input and output an answer. The reply generation process works like this. The decoders hidden state is initialized using the final hidden state from the encoder. The EOS (End-Of-Sentence) token is the first input to the decoder which updates it's hidden state. The reply generation process state is initialized using the final hidden state from the encoder. The EOS token is the first input to the decoder which updates its hidden state. A word is sampled from the last layer, it is fed in as input. The hidden state is updated and a new word is output. This process is repeated in a loop until an EOS token is output or some pre-defined maximum answer length threshold is hit. This process is considered inference. It's the process that a

chatbot model goes through in real-time. Once it has already been trained. The training part works slightly differently. In each decoding step, we use the correct word instead of the generated one as the input. Basically the decoder consumes a correct sequence with the last token removed and the EOS token pre-pended. The goal during training is to maximize the probability of the correct next word on each time step where asking the network to predict the next word sequence providing it with a correct prefix. We minimize the error that most popular optimization strategy in ML i.e. Back-propagation.

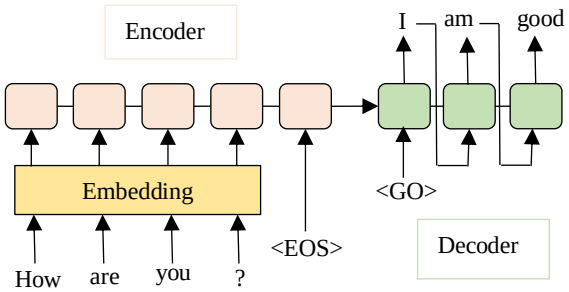

Fig 5: Sequence-to-sequence

The problems identified were,

1) Generic Responses
2) Inconsistent Responses (i.e. asking the same question twice and getting two different responses)

The authors used speaker IDs for each utterance in order to judge an answer not just based only on the encoders state but also on speaker embeddings.

When it comes to selective model, instead of meeting the probability of a certain reply given a certain context, they learn a similarity function where a reply is one of the elements in a pre-defined pool of possible answers. The network will take answers and a possible reply as inputs and will return the confidence of how appropriate they are to each other. Two NNs can be used here and they can be of any given type. The first NN is used for the context, second one for the reply these NN will take their inputs and embedded then into a vector representation then the similarity between them (context_vector and reply_vector) is computed using cosine similarities. During inference it can simply calculate given context and all possible answers to chose the one with maximum similarity. In order to train the model they used triplet loss which includes the context, the correct reply and the wrong reply. By minimizing this flaw they concluded that the similarity function in a ranking way where absolute values aren't informative.

PAPER II: Deep Learning Network for Extreme Multi-Label Text Classification[9]

Extreme Classification is a rapidly growing research area focusing on multi-class and multi-label problems, involving an extremely large number of labels. It has also opened up a new paradigm for ranking and recommendation by reformulating them as multi-label. Such reformulations have led to significant gains over traditional collaborative filtering and context-based recommendation techniques. They have been consequently been deployed in many real-world applications in the industry. They proposed a technique called Extreme Multi-label Classification (XMTC). Two key challenges they dealt with were,

1) Scalability (it is linear i.e. larger the dataset the slower the hidden layers work)
2) Data Sparsity (i.e many labels only appear in very few docs.)

They proposed a CNN called the XML-CNN (Extreme Multi-Label Convolution Neural Network), which was inspired by KimCNN which is a common CNN for text analysis and the strongest method in multi-text classification.

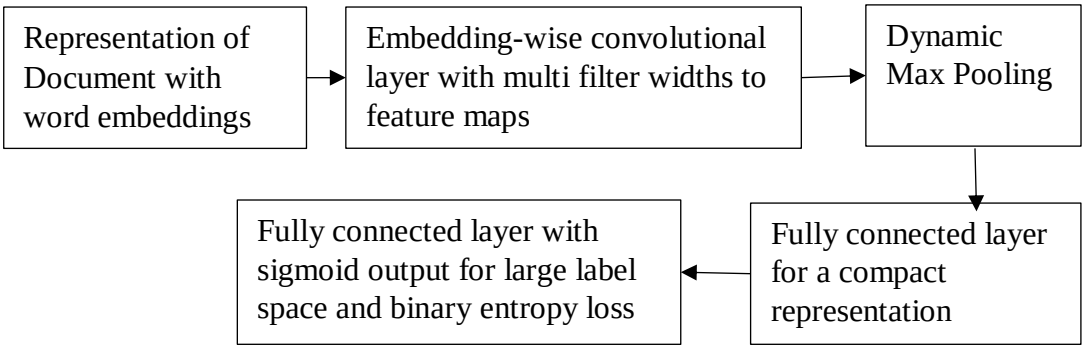

Fig. 6: Working of XML-CNN

Unlike the traditional CNN, here they swipe through the convolution filter in the word-embedding dimension (in an image it is the spatial dimension) because it helps capture salient features. In the feature map they did adaptive max-pulling where they got 2-3 significant features. With low rank factorization they were able to lower the output dimension. They used Cross-entropy over softmax in their model which favored only one label and pushed other labels to zero. This wasn't a good choice which caused their learning curve fall.

PAPER III: Dynamic memory networks for question answering [10]

Dynamic Memory network (DMN) architecture defines two types of memory,

1) Semantic
2) Episodic

The first module, the input module is a GRU that runs on a sequence of word vectors. A GRU cell is kind of like an LSTM cell but its more computationally efficient since it only has two gates and it doesn't use a memory unit. The two gates control when its context is updated and when its erased. And the hidden state of the input module represents the input process, so far in vector. It outputs hidden states after every sentence and these outputs are called facts because they represent the essence of what is fed.

$$h_t = GRU(X_t, h_{t-1})$$

where,

$\boldsymbol{X_t}$ = word vector

$\boldsymbol{ht_{-1}}$ = previous timestep vector

Given a word vector and the previous timestep vector, compute the current timestep vector.

$$U_i = \sigma(W^{(u)}x_i + U^{(u)}h_{i-1} + b^{(u)})$$

The update gate is a single layer NN. They summed up the matrix multiplications and a biased term. And then the sigmoid reduces it to a list of values between 0 and 1(the output vector)

$$r_i = \sigma(W^{(r)}x_i + U^{(r)}h_{i-1} + b^{(r)})$$

It is done twice with different sets of weights

$$h_i = \sigma(W^{(r)}x_i + U^{(r)}h_{i-1} + b^{(r)})$$

Then they used a reset gate that will learn to ignore the past timesteps when necessary. E.g. if the next sentence has nothing to do with those that come before it. The update gate is similar in that it can learn to ignore the current timestep entirely. Maybe the current sentence has nothing to do with the answer. Whereas previous ones did. The Question module, it processes the question word by word and outputs a vector using the same GRU at the input module and the same weights. We can encode both of them by creating embedding layers for both of them. Then we will create an episodic memory representation for both. The motivation for this came from the hippocampus function of our brain. It's able to retrieve temporal states that are triggered by some response like a sight or a sound.

Both the fact and question vectors that are extracted from the input enter te episodic memory module. It's composed of two nested GRU's. The inner GRU generates what are called

episodes. It does this by passing over the facts from the input module. When updating its inner state, it takes into account the output of an attention function on the current fact.

The attention function gives a score between zero and one to each fact. And so the GRU ignores facts with low scores. After each full pass on all the facts, the inner GRU outputs an episode which is then fed to the outer GRU. The reason we need multiple episodes is so our model can learn what part of a sentence it should pay attention to after realizing after first pass that something else is important. With multiple passes they could gather increasingly relevant information. Hence they concluded with their work that dynamic memory networks are the state of the art in Q&A systems by using both semantic and episodic memory inspired by the hippocampus.

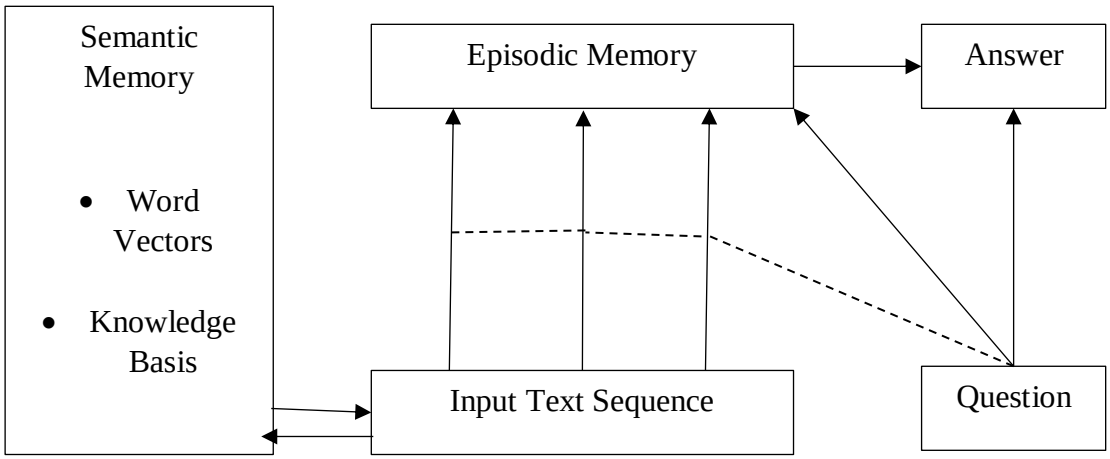

Fig.7: Question Module

PAPER IV: The ChatBot Feels You – A Counseling Service Using Emotional Response Generation[1]

The psychiatric counseling can affect to the user's life directly. Studies have shown that since chatbots have intervened in counseling services there has been a significant change to the patient's habits. E.g. the drinking habit of a patient has notably been low has compared to counselling without a chatbot. But as chatbots fail to show empathy, they fall short here. They proposed a Chabot application which can provide mental healthcare counseling service based on NLP and emotion recognition methods in chat assistant platform which consist of the user sensitive emotion and context extraction. They collected the data from the public media such as drama and radio scripts and SNS dialogues which contains manually descripted emotions.

They developed a deep interaction model that recognizes complex and long term emotions in various conversations continuously. Just as a human counselor needs to learn from many interaction and communication to react counselee properly, the emotional intelligent assistant should communicate and learn opinions and emotions with many people. Four steps are followed like

1) Feature Extraction
2) Response Decision
3) Generating Sympathetic response
4) Generating informative response

Since every user has a different linguistic ability. They normalized data into a fixed length vector representation. Using GRU-based sentence analysis, the model extract the utterance intention to the representation. The similarity analysis is done with cosine similarity measuring. They collected the data from the public media such as drama and radio scripts and SNS dialogues which contains manually descripted emotions.

PAPER V: A Chatbot Using LSTM-based Multi-Layer Embedding for Elderly Care[6]

According to demographic changes, the services designed for the elderly are becoming more needed than before and increasingly important. In this study, we collected the MHMC chitchat dataset from daily conversations with the elderly. Since people are free to say anything to the system, the collected sentences are converted into patterns in the preprocessing part to cover the variability of conversational sentences. Then, an LSTM-based multi-layer embedding model is used to extract the semantic information between words and sentences in a single turn with multiple sentences when chatting with the elderly. Finally, the Euclidean distance is employed to select a proper question pattern, which is further used to select the corresponding answer to respond to the elderly. For performance evaluation, five-fold cross-validation scheme was employed for training and evaluation. Experimental results show that the proposed method achieved an accuracy of 79.96% for top-1 response selection, which outperformed the traditional Okapi model. In the training phase, the proposed system is divided into three parts, which are preprocessing, sentence-based representation, and response selection. The MHMC chitchat database was collected from Chinese daily dialogues, which contain multiple MR pairs. In pre-processing, each message turn in a dialogue will be split into sentences and words. Then, message patterns are generated by replacing the words in the message with manually

pre-defined tags. Message patterns can reduce the variability of the messages in the database. Therefore, their system can handle complex dialogues. Finally, each message pattern is converted into a vector representation based on the method of word/sentence embedding. Then, they used the long short-term memory (LSTM) model to convert the content of sentences into sentence-based representation vectors. Finally, Euclidean distance is employed to select a proper question pattern and its corresponding response. In the test phase, the input message is first segmented into words, converted into patterns and represented using word embedding. Next, sentence representation embedding is used to extract the sentence representation vector of the input message. In the end, they calculated the Euclidean distance to select the appropriate message pattern and choose the corresponding response as the final response sentence. This study proposed a chatbot for elderly care, which can accompany and chat with the elderly people to reduce their loneliness.

CHAPTER 6

TECHNICAL SPECIFICATIONS

From our literature survey we concluded that the problems of the NN models is that the relations made between words are not being interpreted properly. But from the seq-to-seq[9] approach they were successfully able to make the model learn from the word embeddings fed to it. Another problem faced in the learning process was the loss of information due to exploding/vanishing gradient problem. LSTMs[8] have proved to solve this problem. The major problem was on what is most relevant to be remembered and what must be forgotten. Since the Dynamic Memory model[10] was good for smaller datasets, we would propose a model that used GRU instead of an LSTM and merge it with two seq-to-seq models for the learning process. Similar to the Dynamic Memory model we will also use one seq-to-seq model as the input module and the other as the question model.

6.1 Environmental Setup

Hardware Setup:

Personal Desktop with minimum 8GB ram configuration and alteast 500GB internal memory is the minimum requirement.

Software Setup:

Two approaches we could follow is either have all the softwares and packages installed manually or use Google Colab that already has everything setup and requires only internet.

Google Colab is a free cloud service that also supports GPU. It already supports popular libraries like Keras, Tensorflow, PyTorch and OpenCV. The main requirement for this is good speed internet and a google account. Any notebook created on colab will automatically be saved on your google drive linked to the google account used at sign in. The main reason of choosing this service was because to run any ML model a minimum of 8GB RAM is required, but isn't sufficient when training large datasets. It would train but would take several hours to do so. Google Colab service allows us to code hassle free as it already has the package dependencies supported.

In the manual setup process our personal desktop would require additional installations of the python setup along with its UI application(Python Anconda). This won't be enough as we will have to manually install all the packages(Keras) used in the model creation, which is a tedious task.

6.2 Python Packages

Keras is a library used for creating deep learning models. Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. In this project to create S2S and GRU model it has been used.

6.3 bAbI Dataset

It provides a set of tasks. Each task contains a training and testing set with the intention that a successful model performs well on test data. Each task provides a set of training and test data, with the intention that a successful model performs well on test data. The data itself is produced using a simple simulation of characters and objects moving around and interacting in locations. There are 20 types of use-cases provided in this dataset.

Task - 1: Single Supporting Fact

Consists of questions where a previously given single supporting fact, potentially amongst a set of other irrelevant facts, provides the answer. E.g.

"Mary travelled to the office."

Q: Where is Mary?".

A: Office

Task - 2: Two Supporting Fact

A harder task is to answer questions where two supporting statements have to be chained to answer the question. E.g.

"John is in the playground"

"John picked up the football"

Q: Where is the football?

A: John

Task - 3: Three Supporting Fact

It has three supporting statements.

Task - 4: Two Argument Relations

Extreme cases where sentence features are reordered words. E.g.

"What is north of the bedroom?"

"What is the bedroom north of?"

Task - 5: Three Argument Relations

Cases to differentiate three separate arguments.

Task - 6: Yes/No Questions

Cases that force the model to answer true/false type questions. E.g.

Q: ***"Is John in the playground?".***

A: True

Task - 7: Counting

Cases to perform simple counting operations, by asking about the number of objects with a certain property. E.g.

Daniel picked up the football. Daniel picks up the football.

Daniel dropped the football. Daniel drops the newspaper.

Daniel got the milk. Daniel picks up the milk.

Daniel took the apple. John took the apple.

Q: How many objects is Daniel holding?

A: two

Task - 8: Lists

Cases to produce a set of single word answers in the form of a list. E.g. for the above story,

Q: What is Daniel holding?

A: milk, football

Task - 9: Simple Negation

Provides cases with the simplest forms of negation, that of supporting facts that imply a statement is false

Sandra travelled to the office.

Fred is no longer in the office.

Q: Is Fred in the office?

A: no

Q: Is Sandra in the office?

A: yes

Task - 10: Indefinite Knowledge

Cases to model statements that describe possibilities rather than certainties.

John is either in the classroom or the playground.

Sandra is in the garden.

Q: Is John in the classroom?

A: maybe

Q: Is John in the office?

A: no

Task - 11: Basic Coreference

Cases of coreference, that of detecting the nearest referent. E.g.

Daniel was in the kitchen.

Then he went to the studio.

Sandra was in the office.

Q: Where is Daniel?

A: studio

Task - 12: Conjunctions

Cases referring to multiple subjects in a single statement

Mary and Jeff went to the kitchen.

Then Jeff went to the park.

Q: Where is Mary?

A: kitchen

Q: Where is Jeff?

A: park

Task - 13:Compound Coreference

Tests coreference in the case where the pronoun can refer to multiple actors.

Daniel and Sandra journeyed to the office.

Then they went to the garden.

Sandra and John travelled to the kitchen.

After that they moved to the hallway.

Q: Where is Daniel?

A: garden

Task - 14:Time Reasoning

Cases to understand the use of time expressions within the statements.

In the afternoon Julie went to the park.

Yesterday Julie was at school.

Julie went to the cinema this evening.

Q: Where did Julie go after the park?

A: cinema

Q: Where was Julie before the park?

A: school

Task - 15: Basic Deduction

Cases of basic deduction via inheritance of properties.

Sheep are afraid of wolves.

Cats are afraid of dogs.

Mice are afraid of cats.

Gertrude is a sheep. Greg is a swan.

Q: What is Gertrude afraid of?

A: wolves

Task - 16:Basic Induction

Cases of basic induction via inheritance of properties.

Lily is a swan.

Lily is white.

Bernhard is green.

Q: What color is Greg?

A: white

Task - 17: Positional Reasoning

Cases to test spatial reasoning, based on the classical SHRDLU system by asking questions about the relative positions of colored blocks.

The triangle is to the right of the blue square.

The red square is on top of the blue square.

The red sphere is to the right of the blue square.

Q: Is the red sphere to the right of the blue square?

A: yes

Q: Is the red square to the left of the triangle?

A: yes

Task - 18: Size Reasoning

Cases that require reasoning about the relative size of objects and is also inspired by Winograd schema challenge.

The football fits in the suitcase.

The suitcase fits in the cupboard.

The box is smaller than the football.

Q: Will the box fit in the suitcase?

A: yes

Q: Will the cupboard fit in the box?

A: no

Task - 19: Path Finding

Cases to find the path between locations.

The kitchen is north of the hallway.

The bathroom is west of the bedroom.

The den is east of the hallway.

The office is south of the bedroom

Q: How do you go from den to kitchen?

A: west, north

Q: How do you go from office to bathroom?

A: north, west

Task - 20: Agent's Motivations

Cases on why an agent performs an action.

John is hungry.

John goes to the kitchen.

John grabbed the apple there.

Daniel is hungry.

Where does Daniel go?

A: kitchen

Why did John go to the kitchen?

A: hungry

CHAPTER 7

PROPOSED WORK

In our proposed work we will use sequential models and feed then with the fAbI tasks dataset. But the dataset requires some cleaning and re-modeling. Initially the data would look something like this - **John goes to the kitchen. John grabbed the apple there. Daniel is hungry. Where does Daniel go? kitchen 112** We need to break this down to stories, questions and answers. When we clean this data we will get something like **[{"John goes to the kitchen.", "John grabbed the apple there.","Daniel is hungry."}, "Where does Daniel go?", "kitchen", 112].** Here the story is on the zeroth index, the question on the first index, the answer on the second index and the speaker ID on the third index.

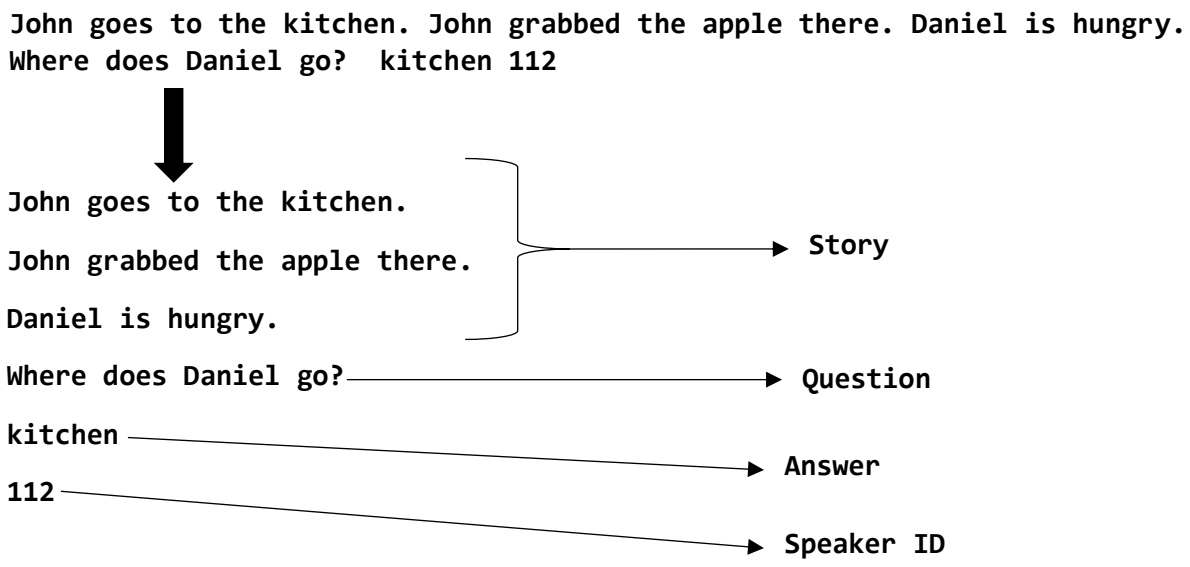

Fig.1: Recurrent Neural Networks(RNNs)

After the cleaning process the data is split into training and testing sets and each are converted into vectors. Three NN models are used, Two sequential and one GRU. The first sequential model is given the story while the second sequential model is given the question. The dot product of both the outputs are generated and fed to softmax activation function as this data has an exponential growth. Usage of sigmoid here is to calculate the probabilities of each word vector to another. This will help output a more relevant work that is close to the query. Its output is then added to the question model's output again to generate a response. Then we concatenate this response with question model output and this answer and fed it to the GRU..

The GRU model will train over the inputs i.e. stories and questions. To validate whether the learning is successful we will check the accuracy through the testing data.

In our work there is a validation model and our GRU model as shown in the figure.

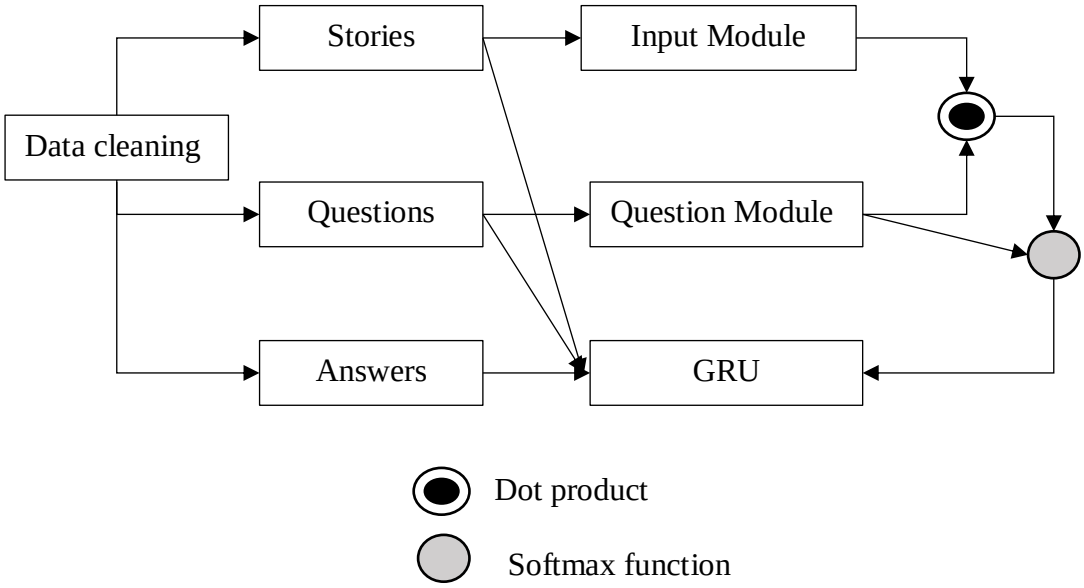

Fig.8: Block Diagram

Pseudo Code:

1) Load dataset from `babi-tasks-v1-2.tar.gz` downloaded from link[19].
2) Data Cleaning:
   a. Divide the data into training set of 10000 entries and testing set of 1000 entries.
   b. Break the dataset into groups of stories, questions, answers and speaker id
   c. Extract stories, questions and answers
3) Convert the stories into tokens and then into one-hot encodings
4) Feed the vectors of stories to the first sequence model that will be the input module
5) Input module output is stored as the `encoded_stories`
6) Convert the questions into token and then into one-hot encodings
7) Feed the vectors of questions to the second sequence model that will be the question module
8) Questions module output is stored has `encoded_questions`
9) The dot product of `encoded_stories` and `encoded_questions` is calculated
10) The dot product is added to the `encoded_questions` and fed as input to GRU for training purpose

For calculating the accuracy the Testing phase output i.e., the predicted output is compared to the actual output we originally got from the dataset.

For example, for the “agents_motivation_10k” tasks we will get an accuracy of 97%

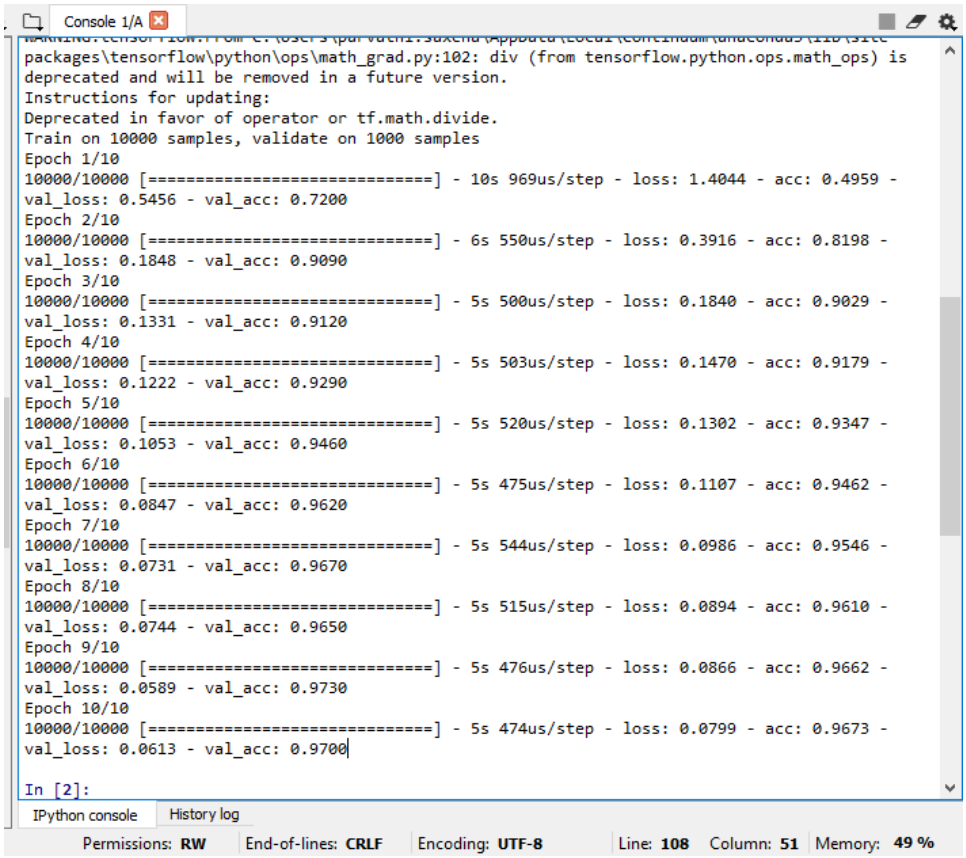

Fig 9: Output of Agent’s Motivation task

Encoding look something like this,

```
Out[8]:
array([[ 0,  0,  0, ..., 22, 33,  1],
       [ 0,  0,  0, ..., 22, 13,  1],
       [ 0,  0,  0, ..., 31, 25,  1],
       ...,
       [ 0,  0,  0, ..., 22, 13,  1],
       [ 0,  0,  0, ..., 10, 32,  1],
       [ 0,  0,  0, ..., 31, 16,  1]])
```

Training/Testing set originally look like this after cleaning,

```
(['Yann','is','hungry','.','Jason','is','tired','.','Jason','travelle
d','to','the','bedroom','.','Yann','travelled','to','the','kitchen','
.','Sumit','is','bored','.','Yann','took','the','apple','there','.','
```

```
Sumit','travelled','to','the','garden','.'], ['Why', 'did', 'sumit',
'go', 'to', 'the', 'garden', '?'],'bored')]
```

7.1 RESULTS

We compared our results for all cases with the LSTM model and Dynamic Memory Model. The LSTM and Dynamic Memory results were recorded from the previous works data. Each task had 10000 training stories and 1000 testing stories.

Challenge Type	Vocab size	Story Max length	Query Max length	LSTM	GRU	DM
single_supporting_fact_10k	22	68	4	50	93	100
two_supporting_facts_10k	36	552	5	20	69	100
three_supporting_facts_10k	37	1875	8	20	21	20
two_arg_relations_10k	49	68	5	61	63	71
three_arg_relations_10k	33	112	4	70	85	83
yes_no_questions_10k	38	156	6	48	60	47
counting_10k	46	321	7	49	79	68
lists_sets_10k	50	358	5	45	75	77
simple_negation_10k	27	74	6	64	63	65
indefinite_knowledge_10k	28	89	6	44	50	59
basic_coreference_10k	29	76	4	72	73	100
conjunction_10k	23	87	4	74	77	100
compound_coreference_10k	29	86	4	94	95	100
time_reasoning_10k	30	116	7	27	39	99
basic_deduction_10k	28	44	6	21	49	74
basic_induction_10k	20	41	5	23	49	27
positional_reasoning_10k	23	24	12	51	62	54
size_reasoning_10k	23	169	10	52	90	57
path_finding_10k	35	40	11	8	100	0
agents_motivation_10k	42	69	8	91	98	100

Table 1: Accuracy Comparison between LSTM, GRU and Dynamic Memory

From the above results we can say that GRU was able to perform well in 50% of the tasks as compared to the other models.

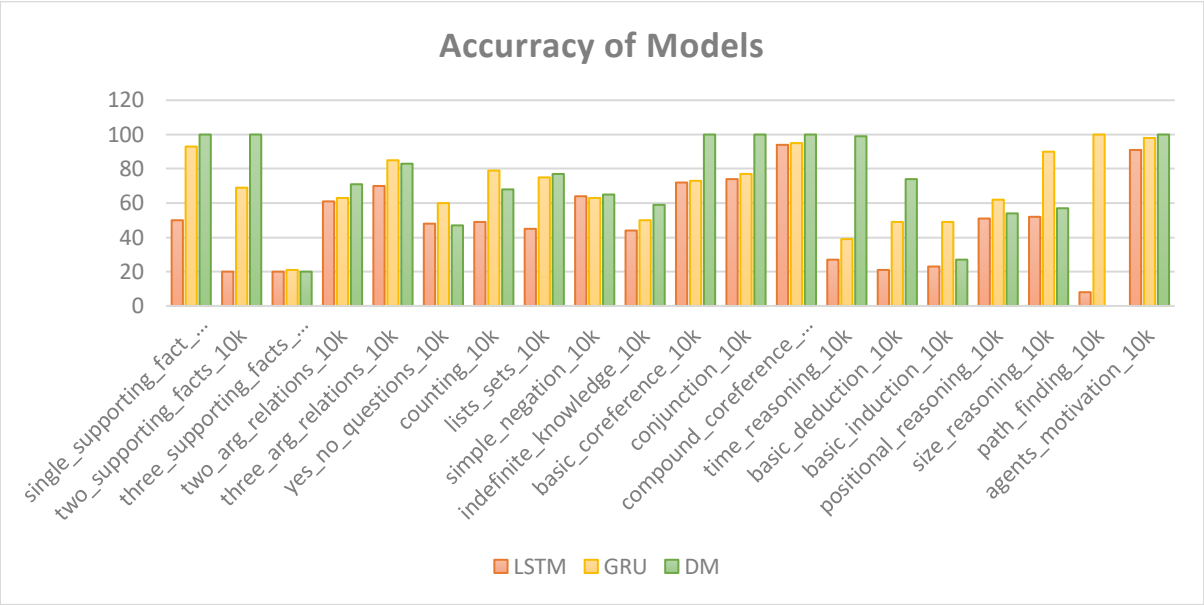

Figure 10: Bar Chart on Accuracy of Models

CHAPTER 8

CONCLUSION

Here we have made the first attempt to creating a smarter agent for chatbot applications. With the results found we can say that GRU is a stronger learner but fails in certain scenarios like three argument relations, deduction and time induction. This is because the model fails to match between question and answer, as it requires three-way interaction as also proved in Watson's[18] work. This model has helped us identify the challenges it faces in different conversational scenarios. One of the major challenges it faces is keeping up with a conversation with different statements that show inheritance properties for deducing the response. So our future plan for the model would be to implement some mechanism to induce inference from the stories.

CHAPTER 9

PROJECT WORK SCHEDULE

(a) Aug - Sept: Status – Completed

Background History and Problem Formulation

(b) Oct – Nov: Status – Completed

Literature Review, Research Direction, Getting and processing dataset, writing of the Survey Paper

(c) Dec – Jan: Status – Completed

Gaining Insights from the data, Implementation of the System.

(d) Feb – Mar: Status – Completed

Experiment the implemented system, Analyse and improve the results found in the experiment.

(e) Apr – May : Status – Scheduled

Result and Conclusion, Writing of the Final Paper, Writing of the Thesis.

9.1 Overall Project Status: On Schedule

No	Task	Aug - Sept	Oct-Nov	Dec-Jan	Feb-Mar	Apr-May
1	Background History					
2	Problem Formulation					
3	Literature Review					
4	Research Direction					
5	Getting and Processing Dataset					
6	Writing of Survey paper					
7	Gaining insights from the data					
8	Implementation of the System					
9	Experimentation of Model					
10	Analysis of Results					
11	Simulation and Verification					
12	Result and Conclusion					
13	Writing Thesis and Paper					

Legend

Completed

Table 2: Time Line Chart for Dissertation

CHAPTER 10

PAPER PUBLICATION

© 2019 JETIR January 2019, Volume 6, Issue 1 www.jetir.org (ISSN-2349-5162)

Study of Neural Network Based Approaches to implement Modern Chatbots

Parvathi Saxena
Manipal University
Jaipur,Rajasthan

Samrit Kumar Maity
C-DAC
Pune, Maharashtra

Manish Kumar Nirmal
C-DAC
Pune, Maharashtra

Akhilesh Kumar Sharma
Manipal University
Jaipur, Rajasthan

***Abstract:** With the emerging trend of chat applications across the globe has made chatbots a new technology that has changed the way we do our conversations. It is a technology that brings many advantages with it for businesses that adopt it. Having an interactive chatbot whose narrative is not programmer driven but instead its own would give rise to a even stronger virtual companion. But these chatbot applications have certain shortcomings. The chatbot fails to understand the context of a conversation and looses the trail of it while conversing, or it might fail to process large training sets due to scalability issues. Here we will discuss the popular neural network models and word embedding techniques for building a smarter chatbot. We also address the shortcomings of these approaches so that we can identify the research area that needs attention to create a smart chatbot that can successfully identify the user's query and generate the right and meaningful response. Mostly chatbots fail to truly understand human conversations, making it understand the context of the conversation will be the first step to a better chatbot.*

Keywords: Chatbots; LSTM(Long-Short Term Memory); deep learning; RNN (Recurrent Neural Network); CNN(Convolutional Neural Network); ChatBot; Conversational agent; AI (Artificial Intelligence); NLP (Natural Language Processing); ML (Machine Learning);

I. INTRODUTION

2016 was the year when chatbots came into the lime light. WhatsApp, Messenger, WeChat, QQ Mobile and a bunch of other popular messaging platforms now host chatbots that developers have built for them. Companies are increasingly engaging with their customers through chatbots. A smart chatbot could be taught of as a service that is driven by artificial intelligence and its narrative is self-driven. Various applications of chatbots other than chat support include booking, counseling, monitoring public chatrooms etc.

Earlier a chatbot's narrative was programmer driven, in the sense that the programmer would have to predict all possible inputs for the bot and make a list of those responses for every input. All of this failed as it was impractical for the programmer to predict all the possibilities. Before deep learning hit the scene, the classical approach to designing a chatbot included, (i) taking a labeled dataset and tokenize the text, (ii) create BoW (Bag of Words), (iii) convert the BoW into TF-IDF matrix(to reduce weight of common words), and (iv) train a classification algorithm(like Naïve Bayes. SVM, Random Forest) on this matrix. Using machine learning approach will lose the word ordering information[20]. E.g. "Pizza Hut is better than Dominos" & "Dominos is better than Pizza Hut" don't mean the same but will have same representation.

It suffers from data sparsity causing it hard to find a decision boundary. But with Neural Networks there has been a shift in this approach. Initially, we will encode all words into vectors and then feed these vectors as input to the neural network so as to train them for classification. What is to be kept in mind is that language is ambiguous and this imposes a challenge on the chatbot application to understand the interpretation of the word(i.e. context). To keep a conversation going the bot is required to stay in touch with the conversation's person/object. This would require the bot to successfully identify the object. But because of ambiguity, the conversation's object may change i.e. the user(human aspect) will change the conversation about something else. Understanding this change is also necessary. In the neural network approach since words are encoded into vectors, feeding them as input for processing would lead to either vanishing/exploding gradients problem. And these can cause the bot to falsely miss a word. Another issue would be processing very large datasets. Conversations today have evolved over the years. There are typos created in texting and tweeting like WRU for 'Where aRe yoU'. Millions of tweets and replies to social media posts are generated. Using this data for training would immensely help in training the network. There are two approaches to implementing deep learning in chatbots, (1) taking different components and applying it to each of them. One system is deep learning based used to interpret the language and another one to track the state of a conversation and then another one generates a response. To train this model, each of the separate systems would be trained separately to do its own task. The chatbot would collectively use those results from each system. This is unnecessarily complex to build. A better type (2) is called the "End-to-End" systems. These are chatbots that use one system that is trained on one dataset. There is no assumptions about the use case or the structure of the dialog. It is trained on the relevant data. End-to-End systems are what we should all be striving for. intuitively they make sense and they're starting to outperform all other systems.

The aim is to investigate and compare the alternative approaches that were followed by various researchers to implement smart chatbots in different scenarios.

This comparison will attempt to answer the following question: "What will make an even smarter chatbot?".This paper has been organized in 8 sections i.e. section I gives an introduction to the area of the subject, section II explains popular word embedding techniques, section III explains sequence-to-sequence model, section IV explains the convolution network model, section V explains the long-short term memory, section VI explains gated recurrent unit for text classification, section VII gives the conclusion of the study and section VIII gives the refernces of this study.

II. WORD EMBEDDINGS

Many Deep Neural Network models today are based on word embedding, where each word in the sentence is mapped to a low dimensional vector.Word Embedding is a NLP technique for finding links between words from raw text through vectors. The basic idea is to convert words in a sentence into meaningful vectors(a.k.a semantic hashing). It learns things based on the context. With the context it can find other words with the similar context based on close spatial positions. Consider the following example, if we are looking for a pizza parlor for dinning, with word embedding we could find it by, in the fig.1, we can find pizza serving restaurants nearby and non-pizza serving restaurants far away. The basis is that word embedding is trained in an unsupervised way i.e. no labelled data is there. Data Sparsity is reduced since we are not dealing with 1's and 0's instead we have floating values[5]. But Gehui[4] et. al argued that word embedding cannot express the interpretation of the word in context as it may suffer from exploding or vanishing gradientproblems.

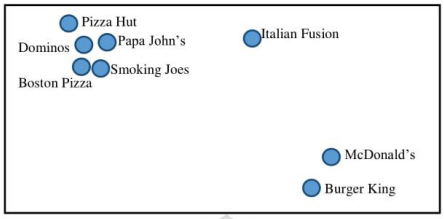

Fig 1: Example for word embedding

Word embedding allows to analyze words mathematically. The high dimensional vectors represent words and each dimension encodes a different property. It is a simple light-weigth neural network that takes a word as a vector. In this technique the words of a sentence are broken down and each word is then converted into a vector. After that it is passed through a neural network to find close words to that word. The output of this network.

A. word2vec

It is a light-weight neural network that takes a word as a vector. In this technique the words of a sentence are broken down and each word is then converted into a vector. After that it is passed through a neural network to find close words to that word. The output of this chain is a set of vectors that are spatially close to the input vector. It is a predictive approach.

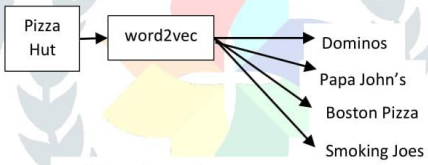

Figure 2: word2vec processing

Consider the example, where the user query is "Find me a pizza parlor like Pizza Hut.". Here each word in the query will be broken down and converted into a vector. With the Pizza Hut vector we will be adequate to find a set of vectors including {'Dominos','PapaJohn's','BostonPizza','Smoking Joes'}. Using this approach enables us to better exploit inter-lexical dependencies between different parts of the input[2].

B. Co-occurrence Matrix

There are other word embedding techniques based on co-occurrence matrix. One common method is to decompose the co-occurrence matrix into smaller matrices. This technique is based on linear algebra and includes breaking the matrix into (say) 2 matrices and the multiplication of these two matrices gives the co-occurrence matrix.

	Pizza Hut	Italian Fusion	Burger King
Pizza Hut	0	3	1
Italian Fusion	3	0	1
Burger King	1	1	0

Table 1: Co-occurrence Matrix

C. Glove

GLObalVEtors(GLOVE)[10] is a count-based approach. A co-occurrence matrix is constructed based on context by words. For each row (i.e. word) it will count the frequently occurring context(i.e. column). Since the number of columns can be large, it factorizes the matrix to get a

lower dimensional matrix which represents words by features. For each row has a feature representation for each word. They can also be trained on large datasets. It performs faster than word2vec.

III. SEQUENCE-TO-SEQUENCE

A Seq2seq model generally consists of two recurrent neural networks. One is the encoder and the other is the decoder. The main objective of this model is to convert a sequence of symbols into a fixed length vector that will help to encode only important information and losing the unnecessary

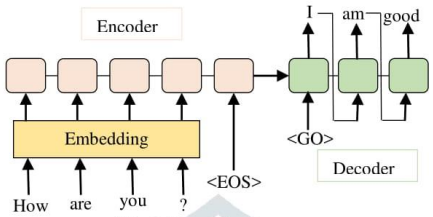

Fig 2: Sequence-to-sequence

information. Each hidden state in the encoder will influence the next hidden state until the final hidden state is achieved. This state is the context (a.k.a thought vector or summary of sequence). With the context, the decoder generates another sequence which is influenced by both the context and previous state. The model processes input as one symbol(word) at a time. Quoc[2] et.al proposed a seq2seq framework that generates simple conversations given a large conversational training dataset. Their model was trained on a large dataset of movie subtitles. Since there was no determining who is talking they considered every sentence as a new question dialogue and the next sentence would the reply dialogue. They used a greedy approach where during inference if the predicted output didn't match the true output then the predicted output would be fed back as an input to predict the next output. They used padding for the input data. So during encoding when the <EOS> symbol is found the decoder is activated. This was the first approach to exploit the encoder-decoder architecture. Their model lacked to ensure consistency and general world knowledge, therefore, failing to keep the conversation going. Existing conversational datasets don't contain all of the world's knowledge and therefore this becomes a primary challenge in building fully data-driven conversational models[2]. Marjan[3] et.al proposed a knowledge-grounded approach where it avoids to learn the same conversational pattern for each distinct entity. It relies on retrieved facts to generate a response. It is made of two GRU networks, unlike Quoc's[2] approach where two LSTM networks are used. They used a separate memory network to pick relevant responses (facts) from the knowledge base. The knowledge base consisted of both world knowledge and conceptual knowledge based on conversations. This was collected from various sources like Wiki, Foursquare, and Twitter. The context generated from the encoder and the facts from the memory network are passed to a softmax function, its output is fed to the decoder to generate the final output(answer).

Various challenges faced by this model include (1) it cannot handle variable length sequences, and (2) large vocabulary size will slow the training process. Common solutions to (1) include (a) padding and (b) bucketing.

1. PADDING
   This technique converts a variable length sequence into a fixed length sequence by appending pads(symbols) into that sequence. Symbols include,
   1) EOS (End Of Sequence)
   2) PAD - Filler
   3) GO – indicator to start decoding
   4) UNK (Unknown) – used when the word is not in the vocabulary

   e.g.

   Question:"FIND ME A PIZZA PARLOR LIKE PIZZA HUT?"
   ⇓
   [PAD, "?", "HUT", "PIZZA", "LIKE", "PARLOR", "PIZZA", "A", "ME", "FIND"]
   ⇓
   length =10
   Answer: "DOMINOS, PAPA JOHN'S, BOSTON PIZZA, SMOKING JOES"
   ⇓
   [GO, "DOMINOS", "PAPA JOHN'S", "BOSTON PIZZA", "SMOKING JOES", PAD, PAD, PAD, PAD, PAD]
   ⇓
   length =10

   The problem faced by this approach is overshadowing actual information i.e. for large sequences we set the length as 100 then for smaller sequences there will be larger number of pads and very less data. This is wastage of memory.

2. BUCKETING
   This technique converts a variable length sequence into buckets of different sizes.

e.g. Buckets are defined as [(5,10), (10,15), (20,25), (40,45)]

Question:"NAME SOME PIZZA PARLORS?"

⇓

["?", "PARLORS", "PIZZA", "SOME", "NAME"]

⇓

length =5

Answer:"PIZZAHUT, DOMINOS, PAPA JOHN'S, BOSTON PIZZA, SMOKING JOES"

⇓

[GO, "DOMINOS", "PAPA JOHN'S", "BOSTON PIZZA", "SMOKING JOES", PAD, PAD, PAD, PAD, PAD]

⇓

length =10

Sequence-to-sequence models must be trained with large datasets but the resulting output may not maintain consistency and fluency[2].

IV. CONVOLUTION NETWORK

Convolutional Neural Network(CNN) model was inspired by animal visual cortex. They are excellent feature extractors in image regardless of position in an image. Generally used in computer vision however recently applied to NLP tasks like text classification and is giving promising results. The selection criteria for text classification is similar to image classification, only difference is that instead of pixel values we have matrix of word vectors[6]. It performs better than statistical models. Collobert[5] et.al first proposed CNN for text. In their work considering a toy problem like given a sentence we will have word embeddings for each word, to convolve over these we require a window size. A simple concatenation is done on these word embeddings to get a single vector. This vector is then multiplied with a weight matrix that has the weights of the convolution layer. This process is followed until the whole sentence is convolved. With the multiple vectors found at the end of convolving the entire sentence, max-pooling is done. Here the maximum value from each vector will be taken to form a single vector (a.k.a most relevant vector of the sentence) that is then fed to a multi-layer perceptron to get the final output. Kim[6] proposed multiple models of CNN for text classification. They were basically (1)CNN-rad which is a baseline model in which the words are randomly initialized and only changed during the training process. (2)CNN-static in which the word vectors are not changed i.e it has pre-trained vectors from word2vec. (3)CNN-non-static model the back-propagation is done on the word vectors too so they change their value when they are trained. (4)CNN-multichannel model uses both static and non-static models. Here the model has two sets of word vectors where each set is a channel to which a filter is applied. Nadeem[7] et.al proposed a CNN for text classification for question-answering domain. The model is based on window sizes (3,4,5) and used 33 filters to roll over the entire sentence matrix reducing it to a low dimensional matrix. The embedded sentences are than padded to make all the word matrices to the same size. Liu[1] proposed a CNN model for extreme multi-label classification called the XML-CNN(eXtreme Multi-Label). They made a small tweak to the existing CNN model, like swiping through the convolution filter in the word-embedding dimension (i.e. the spatial dimension for an image). This helped capture salient features. They compared their model with non-deep learning models like FastXML, SLEEC and PD-Sparse, and deep learning models like FastText, Bow-CNN and CNN-Kim. XML-CNN would scale to extremely large problems, large feature spaces and label spaces. Dauphin[15] et. al gave evidence that gated CNNs used for language modeling tasks outperformed LSTMs. Adel[32] et. al work supported CNN over LSTM/GRU based models for long sentences. In general CNNs are hierarchical is nature as compared to RNNs which are sequential[31].

V. LONG-SHORT TERM MEMORY

With RNN(Recurrent Neural Networks) being used to learn from stored information over extended time intervals proved time-consuming and insufficient, which lead to the discovery of LSTM(Long-Short Term Memory) in 1997[7]. These neurons can train on something over time and remember something from an infinite number of steps back too. Its structure has a more complex neuron. It is similar to an RNN and processes information sequentially. The difference is the operations with the LSTM cells. These neurons have an internal hidden state that can be updated or it can be forgotten with the operations. RNN is good for sequence processing but suffers from short-term memory[11]. LSTM and GRU were created to mitigate this with a mechanism called gates. It has 4 gates namely, (1)cell gate, (2) forget gate, (3)input gate, (4)output gate. The cell gate carries information throughout the sequentially processing. In theory, even information from earlier time steps can be carried all the way to the last time step

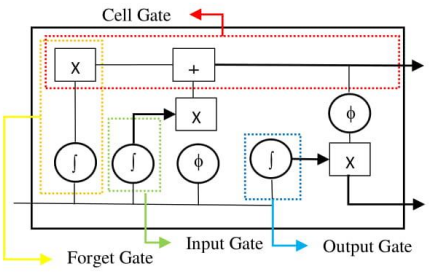

Figure 3: LSTM

and therefore reducing the effects of short-term memory. The gates are just different neural networks that decide which information is allowed on the cell gate and regulate the flow of information from one time step to another. Gates contains sigmoid(∫) and tan(ϕ) activation functions. The ∫ is similar to the ϕ where instead of squishing values to -1 to 1, it will be between 0 and 1. This helps in update and forget operation. The forget gate decides what information should be updated or forgotten. Information from a previous hidden state and information from the current input is passed to ϕ, to get output between 0(forget) and 1(remember). To regulate the network, the previous hidden state and the current state are passed to ϕ, its output is multiplied with ∫ which decides what information to keep. With these information, we calculate the cell state which is then multiplied with the forget vector[9]. The output from the input gate is used to do polarize addition which updates the cell state to new values giving a new cell state. The output gate decides what the next hidden state should be. The hidden state is used for prediction and contains information about previous inputs.

The term LSTM came about because the standard RNN operates in such a way that the hidden state activations are influenced by the other local activations closest to it, which corresponds to "short-term memory". While the network weights are influenced by the computation that takes place over the entire sequence, which corresponds to "long-term memory". This redesign of RNN caused the activation state to also act as weights and preserve information over long periods of time.

The exploding/vanishing gradients problem[17] was analyzed in Hochreiter[7] et. al work. Su[8] et. al proposed a LSTM-based multi-layer embedding model for a Q&A chatbot for the elderly that is used to extract the semantic information between words and sentences. Ney[12] et. al compared LSTM with the traditional RNN using English and French corpus in terms of perplexity and word error rate. LSTM outperformed RNN proving to be a state-of-the art speech recognition system.Felix[14] et. al found that LSTM memory is limited by the number of memory blocks and increasing them homogenously would not help. Some change to the model was necessary for keeping an effective learning process. They also found that where information is kept or thrown, can be triggered by noisy input sequences and therefore the LSTM must be fed with appropriate feature detector.

VI. GATED RECURRENT UNIT

A newer generation of RNN is the GRU which is similar to LSTM. There is no cell gate instead used a hidden state to transfer information. It only has two gates namely (1) reset gate and (2)update gate. The update gate acts similar to the input gate and the forget gate of the LSTM. It decides what information to keep or forget. They are faster than LSTM because they have less tensor operations.

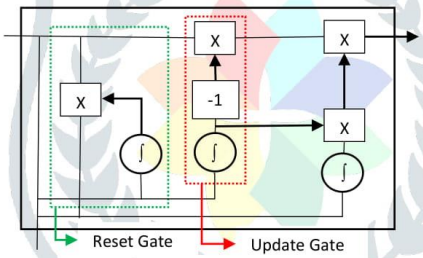

Figure 4: GRU

Cho[20] et. al exploited a RNN encoder-decoder model where one GRU layer was used to model the history of the conversation at the sentence-level while another GRU layer was responsible for modelling each sentence at the word-level. They found that encoding long sequences to fixed-length vectors was problematic for translating. Chung[19] et. al proved that GRU performs better than LSTM in the learning process. As the network uses local context and the current input to regulate the flow of information, they found that removing the local context can reduce the computational complexity of the model but also loss of sequential information. Lai[21] et. al proposed a variant(iCARNN) of the traditional GRU by introducing a new gate called the independent gate. This gate controls the context and input causing the gate computations to simplify. This was unlike the RNN, the gate computations at all time steps can be calculated in parallel. But it retains the sequential nature of RNN. They found that iCARNN either performs the best or very close to the best on the various datasets of different sizes they used. Zhang[22] et.al proposed a hierarchical GRU model that incorporates attention mechanism to better capture important information from a document. This approach effectively reduced the computational complexity and improved the prediction accuracy. Yin[31] et. al found GRU to outperform CNN when fed with larger sentences. Trofimovich[33] et. al work on sentiment analysis of Russian tweets proved GRU performs better than CNN and LSTM models.

VII. CONCLUSION

Chatbots are limited by its own capacity to grasp information and give proper replies. They fail to identify inconsistences or keep up the conversation when the subject changes. It doesn't truly understand the meaning of the sentences it is fed. Using neural networks to improve its learning process has made a tremendous change. Some promising models were discussed in this paper. But there is no right way for selecting the deep neural network model for this problem[31]. We were able to identify that modified RNNs like LSTM and GRUs are providing promising results and successfully learns information to make predictions. LSTMs and GRUs are being used in the state-of-the art deep learning applications like speech recognition, speech synthesis[15], natural language understanding[14] etc. Chatbots have become a new source of interaction between humans and AI.

VIII. REFERENCES

[1] Liu, J., Chang, W.C., Wu, Y. and Yang, Y., 2017, August. Deep learning for extreme multi-label text classification. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 115-124). ACM.

[2] Vinyals, O. and Le, Q., 2015. A neural conversational model. arXiv preprint arXiv:1506.05869. Ghazvininejad, M., Brockett, C., Chang, M.W., Dolan, B., Gao, J., Yih, W.T. and Galley, M., 2017. A knowledge-grounded neural conversation model. arXiv preprint arXiv:1702.01932.

[3] Shen, G., Yang, Y. and Deng, Z.H., 2017. Inter-weighted alignment network for sentence pair modeling. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (pp. 1179-1189).

[4] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P., 2011. Natural language processing (almost) from scratch. Journal of Machine Learning Research, 12(Aug), pp.2493-2537.

[5] Kim, Y., 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

[6] Amin, M.Z. and Nadeem, N., 2018. Convolutional Neural Network: Text Classification Model for Open Domain Question Answering System. arXiv preprint arXiv:1809.02479.

[7] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

[8] Su, M.H., Wu, C.H., Huang, K.Y., Hong, Q.B. and Wang, H.M., 2017, December. A chatbot using LSTM-based multi-layer embedding for elderly care. In Orange Technologies (ICOT), 2017 International Conference on (pp. 70-74). IEEE.

[9] Gers, F.A., Schmidhuber, J. and Cummins, F., 1999. Learning to forget: Continual prediction with LSTM.

[10] Pennington, J., Socher, R. and Manning, C., 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[11] Gers, F.A., Schraudolph, N.N. and Schmidhuber, J., 2002. Learning precise timing with LSTM recurrent networks. Journal of machine learning research, 3(Aug), pp.115-143.

[12] Bengio, Y., Simard, P. and Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2), pp.157-166.

[13] Martens, J. and Sutskever, I., 2011. Learning recurrent neural networks with hessian-free optimization. In Proceedings of the 28th International Conference on Machine Learning (ICML-11) (pp. 1033-1040).

[14] Sundermeyer, M., Schlüter, R. and Ney, H., 2012. LSTM neural networks for language modeling. In Thirteenth annual conference of the international speech communication association.

[15] Dauphin, Y.N., Fan, A., Auli, M. and Grangier, D., 2016. Language modeling with gated convolutional networks. arXiv preprint arXiv:1612.08083.

[16] Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. and Kavukcuoglu, K., 2016. Wavenet: A generative model for raw audio. CoRR abs/1609.03499.

[17] Miller, J. and Hardt, M., 2018. When Recurrent Models Don't Need To Be Recurrent. arXiv preprint arXiv:1805.10369.

[18] Garigliotti, D. and Balog, K., 2018, March. Towards an Understanding of Entity-Oriented Search Intents. In European Conference on Information Retrieval (pp. 644-650). Springer, Cham.

[19] Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

[20] Bahdanau, D., Cho, K. and Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[21] Tran, Q.H., Lai, T., Haffari, G., Zukerman, I., Bui, T. and Bui, H., 2018. The Context-dependent Additive Recurrent Neural Net. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (Vol. 1, pp. 1274-1283).

[22] Shi, Y., Yao, K., Tian, L. and Jiang, D., 2016. Deep LSTM based feature mapping for query classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 1501-1511).

[23] Zhang, L., Zhou, Y., Duan, X. and Chen, R., 2018, March. A Hierarchical multi-input and output Bi-GRU Model for Sentiment Analysis on Customer Reviews. In IOP Conference Series: Materials Science and Engineering (Vol. 322, No. 6, p. 062007). IOP Publishing.

[24] Shen, G., Yang, Y. and Deng, Z.H., 2017. Inter-weighted alignment network for sentence pair modeling. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (pp. 1179-1189).

[25] Lai, T.M., Bui, T. and Li, S., 2018. A Review on Deep Learning Techniques Applied to Answer Selection. In Proceedings of the 27th International Conference on Computational Linguistics (pp. 2132-2144).

[26] Lan, W. and Xu, W., 2018. Character-based Neural Networks for Sentence Pair Modeling. arXiv preprint arXiv:1805.08297.

[27] Csáky, R.K. and Recski, G., Deep Learning Based Chatbot Models.

[28] Choi, Y., Kim, S. and Lee, J.H., 2016, August. Recurrent Neural Network for Storytelling. In Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems, 2016 Joint 8th International Conference on (pp. 841-845). IEEE.

[29] Su, M.H., Wu, C.H., Huang, K.Y., Hong, Q.B. and Wang, H.M., 2017, December. A chatbot using LSTM-based multi-layer embedding for elderly care. In Orange Technologies (ICOT), 2017 International Conference on (pp. 70-74). IEEE.

[30] Bhagwat, V.A., 2018. Deep Learning for Chatbots.

[31] Yin, W., Kann, K., Yu, M. and Schütze, H., 2017. Comparative study of cnn and rnn for natural language processing. arXiv preprint arXiv:1702.01923.

[32] Adel, H. and Schütze, H., 2016. Exploring different dimensions of attention for uncertainty detection. arXiv preprint arXiv: 1612.06549.

[33] Trofimovich, J., 2016. Comparison of neural network architectures for sentiment analysis of russian tweets. In Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference Dialogue 2016, RGGU.

CHAPTER 11

REFERENCES

[1] Dongkeon Lee, Kyo-Joong Oh, Ho-Jin Choi, "The ChatBot Feels You – A Counseling Service Using Emotional Response Generation", 2017, IEEE International Conference on Big Data and Smart Computing (BigComp).

[2] Godson Michael D'silva, Sanket Thakare, Sharddha More, and Jeril Kuriakose, "Real World Smart Chatbot for Customer Care using a Software as a Service (SaaS) Architecture", 2017, International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud).

[3] Bhavika R. Ranoliya, Nidhi Raghuwanshi and Sanjay Singh, "Chatbot for University Related FAQs", 2017, International Conference on Advances in Computing, Communications and Informatics (ICACCI).

[4] Divya Madhu, Neeraj Jain C. J, Elmy Sebastain, Shinoy Shaji, Anandhu Ajayakumar, "A Novel Approach for Medical Assistance Using Trained Chatbot", 2017, International Conference on Inventive Communication and Computational Technologies (ICICCT).

[5] Bhuvan Sharma, Harshita Puri and Deepika Rawat, "Digital Psychiatry – Curbing Depression using Therapy Chatbot and Depression Analysis", 2018, 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT).

[6] Ming-Hsiang Su, Chung-Hsien Wu, Kun-Yi Huang, Qian-Bei Hong and Hsin-Min Wang, "A Chatbot Using LSTM-based Multi-Layer Embedding for Elderly Care", 2017, International Conference on Orange Technologies (ICOT).

[7] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P., "Natural language processing (almost) from scratch", 2011, Journal of Machine Learning Research.

[8] Vinyals, O. and Le, Q., "A neural conversational model". 2015. arXiv preprint arXiv:1506.05869.

[9] Liu, J., Chang, W.C., Wu, Y. and Yang, Y., "Deep learning for extreme multi-label text classification". 2017, August. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 115-124). ACM.

[10] Raghuvanshi, Arushi, and Patrick Chase. "Dynamic memory networks for question answering." (2016).

[11] Shen, G., Yang, Y. and Deng, Z.H.,"Inter-weighted alignment network for sentence pair modeling" , 2017. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (pp. 1179-1189).

[12] Hochreiter, S. and Schmidhuber, J.. “Long short-term memory. Neural computation”, 1997 9(8), pp.1735-1780.

[13] Bhagwat, V.A., “Deep Learning for Chatbots.”, 2018.

[14] Lai, T.M., Bui, T. and Li, S. “A Review on Deep Learning Techniques Applied to Answer Selection”, 2018. In Proceedings of the 27th International Conference on Computational Linguistics (pp. 2132-2144).

[15] Lan, W. and Xu, W.. “Character-based Neural Networks for Sentence Pair Modeling”, 2018. arXiv preprint arXiv:1805.08297.

[16] Csáky, R.K. and Recski, G., “Deep Learning Based Chatbot Models”.

[17] Choi, Y., Kim, S. and Lee, J.H. “August. Recurrent Neural Network for Storytelling”, 2016. In Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems, 2016 Joint 8th International Conference on (pp. 841-845). IEEE.

[18] Sukhbaatar S, Weston J, Fergus R. “End-to-end memory networks”. In Advances in neural information processing systems 2015 (pp. 2440-2448).

[19] https://s3.amazonaws.com/text-datasets/babi_tasks_1-20_v1-2.tar.gz