Spring 2018

Deep Learning for Chatbots

Vyas Ajay Bhagwat
San Jose State University

Deep Learning for Chatbots

A Thesis

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements of the Class

CS298

By

Vyas Ajay Bhagwat

May 2018

The Designated Thesis Committee Approves the Thesis Titled

Deep Learning for Chatbots

by

Vyas Ajay Bhagwat

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2018

Abstract

Natural Language Processing (NLP) requires modelling complex relationships between the semantics of the language. While traditional machine learning techniques are used for NLP, the models built for conversations, called chatbots, are unable to be truly generic. While chatbots have been made with traditional machine learning techniques, deep learning has allowed the complexities within NLP to be easier to model and can be leveraged to build a chatbot which has a real conversation with a human. In this project, we explore the problems and techniques used to build chatbots and where improvements can be made. We analyze different architectures to build chatbots and propose a hybrid model, partly retrieval-based and partly generation-based which gives the best results.

Acknowledgements

I would like to thank Dr. Robert Chun for his continued support and providing me the guidance necessary to work on this project. I would like to thank my advisor Dr. Robert Chun and committee members Dr. Katerina Potika and Mr. Vinit Gaikwad for teaching me core skills needed to succeed and reviewing my project. And finally, I would like to thank my parents for their patience and advice they gave me throughout my life.

TABLE OF CONTENTS

I. INTRODUCTION

In today's world, the way we interact with our digital devices is largely restricted, based on what features and accessibility each device offers. However simple it may be, there is a learning curve associated with each new device we interact with. Chatbots solve this problem by interacting with a user using text autonomously. Chatbots are currently the easiest way we have for software to be native to humans because they provide an experience of talking to another person [1]. Since chatbots mimic an actual person, Artificial Intelligence (AI) techniques are used to build them. One such technique within AI is Deep Learning which mimics the human brain. It finds patterns from the training data and uses the same patterns to process new data. Deep Learning is promising to solve long standing AI problems like Computer Vision and Natural Language Processing (NLP), with Google investing $4.5 million [2] in Montreal AI Lab in addition to a federal AI grant of $213 million [2].

The existing chatbots which are around, like Siri, Alexa, Cortana and Google Assistant face difficulties in understanding the intentions of the user and hence become difficult to deal with. Specifically, these chatbots cannot keep track of the context and suffer in long-ranging conversations. Another shortcoming of these chatbots is they are designed specifically for helping a user with some specific problems, hence restricting their domain. They are unable to make a coherent and engaging conversation between two human beings on popular topics such as recent news, politics and sports.

Through this research project, we first explore different architectures of ANNs for NLP and analyze the existing models used to build a chatbot. We then attempt to gain insight into the questions: *What are Deep Neural Networks and why are they important? How is a chatbot built as of today, what are their limitations and where can we try to improve?* We evaluate some novel implementations and report on their effectiveness. We have analyzed articles which are fundamental to this problem as well as the recent developments in this space. Fig.1 is the conceptual map of the topics in this review.

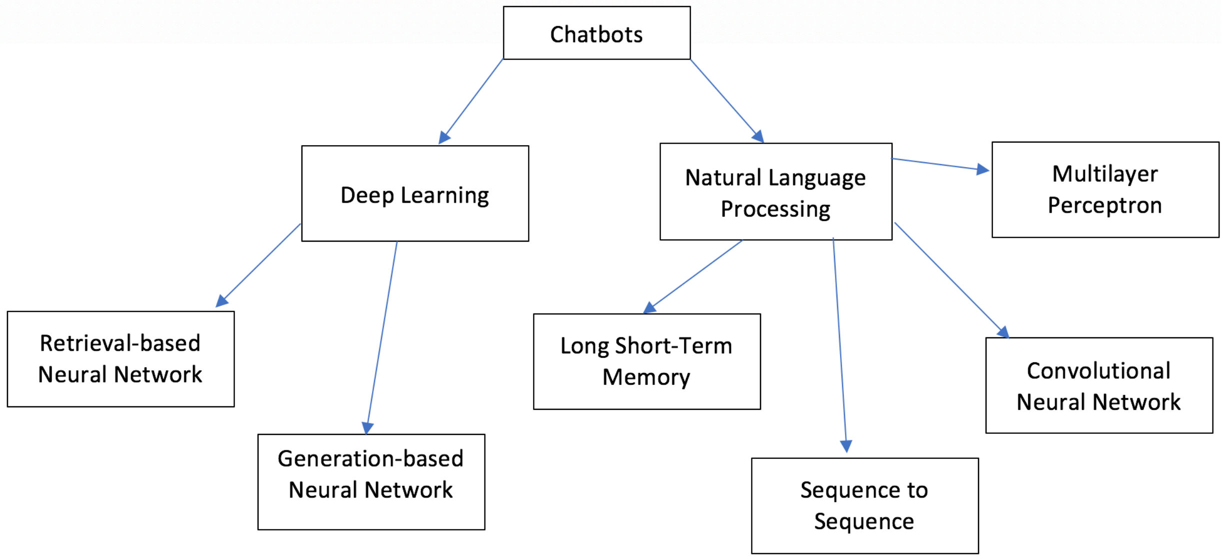

Chatbots
Deep Learning
Natural Language Processing
Multilayer Perceptron
Retrieval-based Neural Network
Generation-based Neural Network
Long Short-Term Memory
Convolutional Neural Network
Sequence to Sequence

Fig. 1. Conceptual map of topics

II. DEEP LEARNING

Goodfellow [3] has categorized AI into three approaches:

A. Knowledge Base

B. Machine Learning

C. Representation Learning

A. Knowledge Base

Most early work in AI can be categorized into this approach. Knowledge based systems have been helping humans to solve problems which are intellectually difficult, but easy for machines. These problems typically are easily represented with a set of formal rules. An example of this could be Mycin which was a tool developed at Stanford University in 1972 to treat blood infections [4]. Mycin was built on rules and was able to suggest a suitable treatment plan to a patient with blood infections. Mycin would ask for additional information whenever required, making it a robust tool for its time.

While Mycin was at par with medical practitioners of the day, it fundamentally operated on rules. These rules were required to be written formally, which was a tough task. Hence, this approach restricts any AI model to one particular, narrow domain, in addition to being difficult to enhance. This might be a reason that none of these projects has led to a major success [3].

B. Machine Learning:

Machine Learning tries to overcome the limitations of hard-coded rules of the Knowledge Base approach to AI. Machine Learning is able to extract patterns from data instead of relying on rules. Simple Machine Learning techniques like linear regression and naïve Bayes methods learn the correlation between features and the output class or value. They have been used to make simple models such as housing price prediction and spam email detection.

Machine Learning techniques allowed machines to understand some knowledge of the real world. The predictions depend on correlation between features and output value. These methods, however, are restricted to the features which are designed by the modeler, which again can be a difficult task. This essentially means that each entity should be represented as a set of features. Consider the problem of face detection as an example. The modeler can represent a face with a set of features such as having a particular shape and structure, but this is difficult to model on a pixel-to-pixel basis.

Another drawback of this approach is that representation of data is extremely important. Fig. 2 shows two different representations of the same dataset using Cartesian co-ordinates and polar co-ordinates. Consider a classification task of separating two entities by drawing a line between them. This task cannot be done on the Cartesian representation but is easy for polar representation. To achieve the best predictions, the modeler has to go through the process of *feature engineering,* which involves representing the data for a model as a preprocessing step. Both knowledge-base and machine learning approaches require us to have substantial domain knowledge and expertise.

One solution to this problem is to use machine learning to discover not only the mapping from representation to output, but also the representation itself [3]. This is where representation learning comes into the picture.

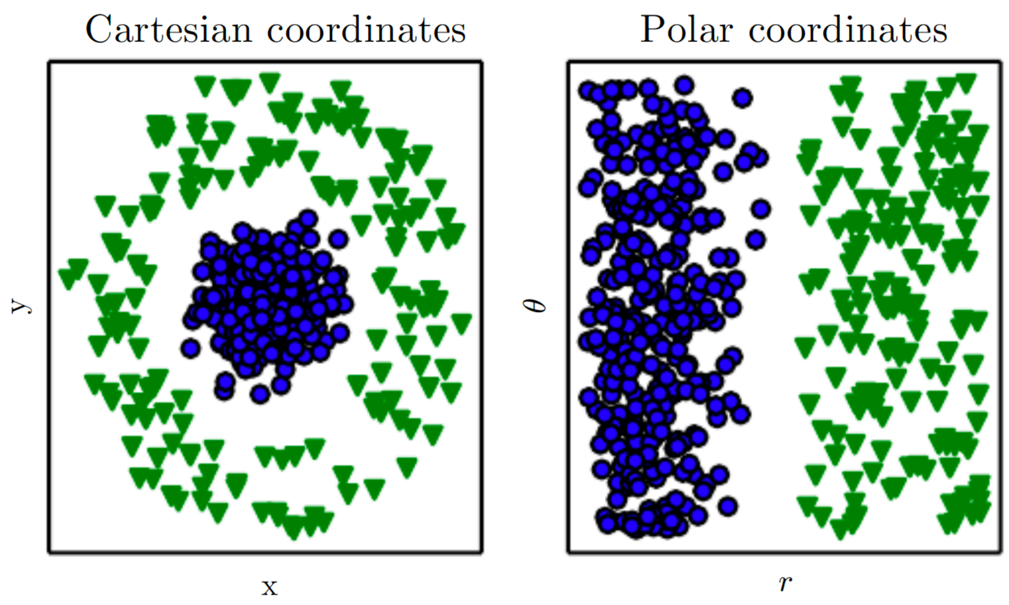


Fig. 2. Cartesian and Polar representation of the same dataset

C. Representation Learning

The need for representation learning comes from the limitations of rigidity of knowledge-base and machine learning approaches. We want the model to be able to learn the representation of data itself. Learned representations often result in much better performance than can be obtained with hand-designed representations [3].

Consider the example of face detection. As humans, we can recognize a face from different viewing angles, different lighting conditions, different facial features such as spectacles or beard. This representation of data is abstract and can be thought of as a hierarchy of simple to complex concepts which allow us to make sense of different data that we encounter. However, this information is almost impossible to model because of the randomness. Deep Learning attempts to overcome this challenge by expressing complex representations in terms of simpler representations [3].

Deep Learning is a subset of representation learning, having multiple layers of neurons to learn representations of data with multiple levels of abstraction [5]. Representation learning models the human brain, with brain neurons analogous to computing units and the strength of connections between the neurons analogous to weights. Deep Learning architecture is similar to an Artificial Neural Network (ANN), but with more hidden layers (hence, more neurons) which allows us to model the more complex functions of our brain. This architecture is shown in Fig. 3.

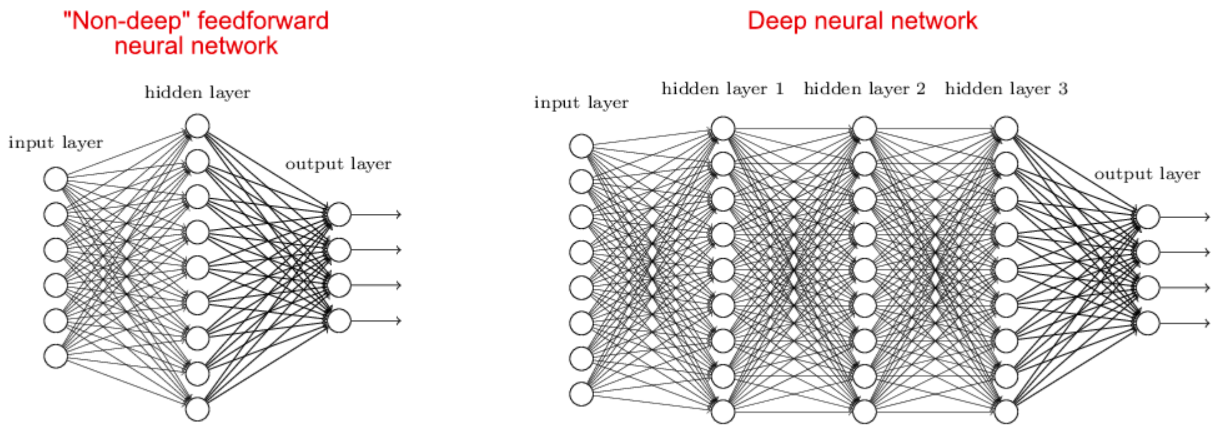


Fig. 3: Deep Learning architecture [6]

Conventional machine learning methods have relied heavily on feature engineering and feature extraction. This is done manually by analyzing the parameters essential to the output via statistical methods and requires a substantial amount of domain knowledge and expertise. Deep Learning allows a machine to learn the representation of data, by transforming an input at one step to a higher, abstracted level [5]. Hence, this step of feature engineering is eliminated in Deep Learning. This allows us to create more generic models which can analyze data at scale.

III. EVOLUTION OF CHATBOTS

A. ELIZA

Chatbot was a term coined my Mauldin [7] to define the systems aiming to pass the Turing Test. The first chatbot which came to public was ELIZA in 1966 [8] and was widely recognized as the first program capable to pass the Turing Test.

ELIZA's model was essentially capturing the input, rephrase it, and try to match keywords with pre-defined responses. As an example, we can consider the below response rule.

utterance: * you are *

answer: What makes you think I am ____?

Here, the * represents a wildcard character. ELIZA recognizes this particular rule from a set of rules, then responds with the corresponding output string. There are some rules written to rephrase the user utterance before matching. Below is an example of this rule.

utterance: You are crazy

answer: What makes you think I am crazy?

In addition to this rule, ELIZA had some responses which were regardless of any context, possibly used when none of the rules were matched. As an example,

utterance: My life is great!

answer: Please go on.

Although ELIZA was created by Weizenbaum to demonstrate how superficial the conversation between a human and a computer is, ELIZA became widely popular because people thought they were talking to a real person [9].

Because ELIZA is a rule-based system, it couldn't have any meaningful conversations with humans. Also, as explained above, it had generic responses to utterances which did not fit into any rules. Hence, one could argue that it was designed only to deceive a human into thinking that it is not a computer program. Though many implementations of ELIZA exist on the internet which can be leveraged to design a new chatbot, scaling this bot is a difficult task. Many chatbots were inspired by this simple rule-based framework, like PARRY and A.L.I.C.E [10].

B. Jabberwacky

Jabberwacky was designed by Rollo Carpenter, a British programmer [11]. It is meant to be a conversational bot making interesting and humorous conversation. It was one of the first chatbots which harnessed AI to learn meaningful human conversations. Jabberwacky won the Loebner prize, an annual competition to test chatbots, in 2005 and 2006 [11]. Jabberwacky was also able to learn continually from a conversation. It can learn new languages, concepts and facts just by constant interactions.

The creators have maintained that Jabberwacky is not based on any artificial life model (no Neural Networks or Fuzzy Logic) and is based purely on heuristics-based technology. It does not have any rules and relies entirely on context and feedback. Since it has been online from 1997, it has learned new languages as well.

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

A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) is a popular free chatbot developed in 1995 by Dr. Richard Wallace. It is inspired by ELIZA and won the Loebner prize in January 2000 [12]. It was first implemented in SETL, a language based on set theory and mathematical logic. This was known as Program A. There were a few contributors at this time until 1998 when it migrated to Java. Dr. Wallace also implemented AIML (Artificial Intelligence Markup Language), an XML dialect for creating chatbots. A.L.I.C.E was written in AIML. In 1999, a Program B was launched where more than 300 developers joined the effort to write A.L.I.C.E and AIML made a transition to fully-compliant XML grammar. The Program B led A.L.I.C.E to win the Loebner prize in 2000. Program C was launched after Jacco Bikker created the first C and C++ based implementation of AIML. This led the way for many different C/C++ threads for the Alicebot engine. Finally, Program D led the transition of A.L.I.C.E from pre-Java 2 to add many features and to make use of features like Swing and Collections.

D. Commercial Chatbots

The bot SmarterChild was launched in 2001 [13] as a service which could answer fact-based questions like *"What is the population of Indonesia?"* or more personalized questions like *"What movies are playing near me tonight?"* [14]. Before being taken down in 2008, it was a popular chatbot used by thousands of people every day.

IBM Watson was developed in 2006 specifically to compete with the champions of the game of Jeopardy! [13]. Watson won the competition in 2011. Since then, IBM Watson offers services to build chatbots for various domains which can process large amounts of data. IBM calls this "Cognitive Computing", claiming to use techniques used by humans for understanding data and reasoning.

The launch of Siri in 2010 as an intelligent virtual assistant paved the way for other virtual assistants like Google Now (2012), Cortana (2015) and Alexa (2015). All of these assistants can help answer questions based on the web and help access a mobile device easily. Recently, they have enhanced capabilities like Image-based search.

Since all of these technologies are commercial, the details of implementation are not available.

IV. NEURAL NETWORKS FOR NATURAL LANGUAGE PROCESSING (NLP)

A. Multilayer Perceptron (MLP)

A Multilayer Perceptron (MLP) is a feedforward network with one input layer, one output layer, and at least one hidden layer [15]. To classify data which is not linear in nature, it uses non-linear activation functions, mainly hyperbolic tangent or logistic function [16]. The network is fully connected, which means that every node in the current layer is connected to each node in the next layer. This architecture with the hidden layer forms the basis of deep learning architecture which has at least three hidden layers [5]. Multilayer Perceptron is used for speech recognition and translation operations of NLP. Fig. 4 shows a MLP with one hidden layer.

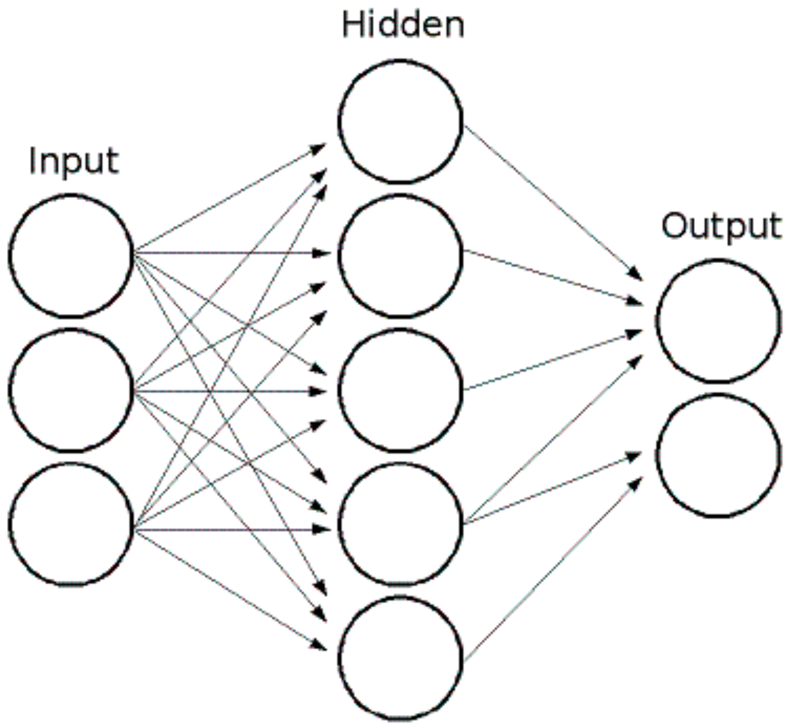


Fig. 4. Multilayer Perceptron with one hidden layer [17]

B. Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a variation of MLP with at least one convolutional layer [16]. The convolutional layer reduces the complexity of the network by applying a convolution function on the input and passing the output to the next layer, analyzing a part of the data (sentence/image) at a time. Since the complexity is reduced with CNN, the network can be much deeper and handle more complex data.

CNNs are popular for computer vision tasks [18]. However, Collobert et al. [19] used CNNs for NLP tasks, such as part of speech (POS) tagging and semantic analysis which are the preprocessing steps for any NLP algorithm. They also proposed a general-purpose CNN architecture to perform all the NLP related tasks at once, which generated interest in CNN for NLP. Fig. 5 shows a convolution operation in CNN.

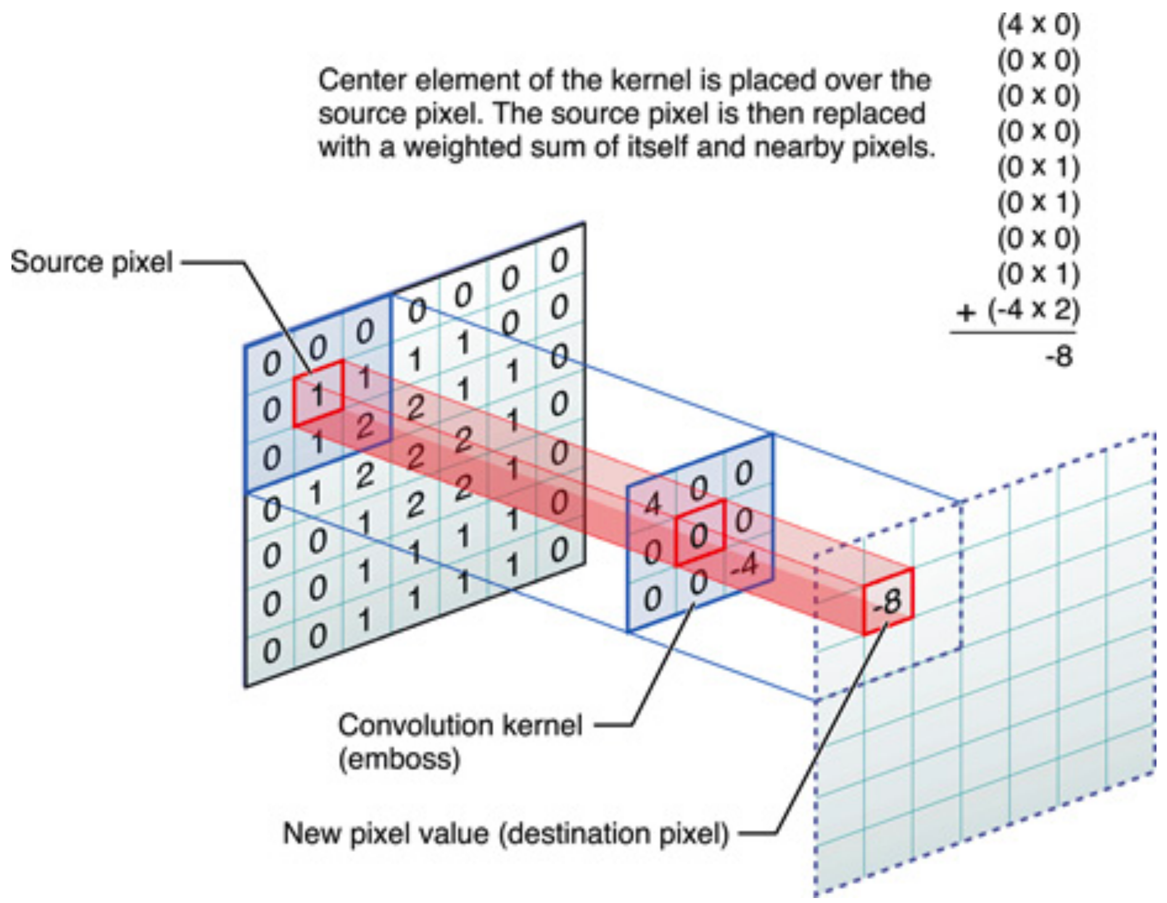

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)

Fig. 5. A convolution operation with convolution kernel, also known as a filter. The filter sequentially goes around all the source pixels and a scalar product is calculated, thus reducing the size of the matrix. [20]

C. *Recurrent Neural Network (RNN)*

A Recurrent Neural Network (RNN) is designed to preserve the previous neuron state. This allows the neural network to retain context and produce output based on previous state [16]. This approach makes RNNs desirable for chatbots as retaining context in a conversation is essential to understand the user. RNNs are extensively used for NLP tasks such as translation, speech recognition, text generation and image captioning [21] [22] [23]. Fig. 6 shows the comparison between RNN architecture and vanilla feedforward network.

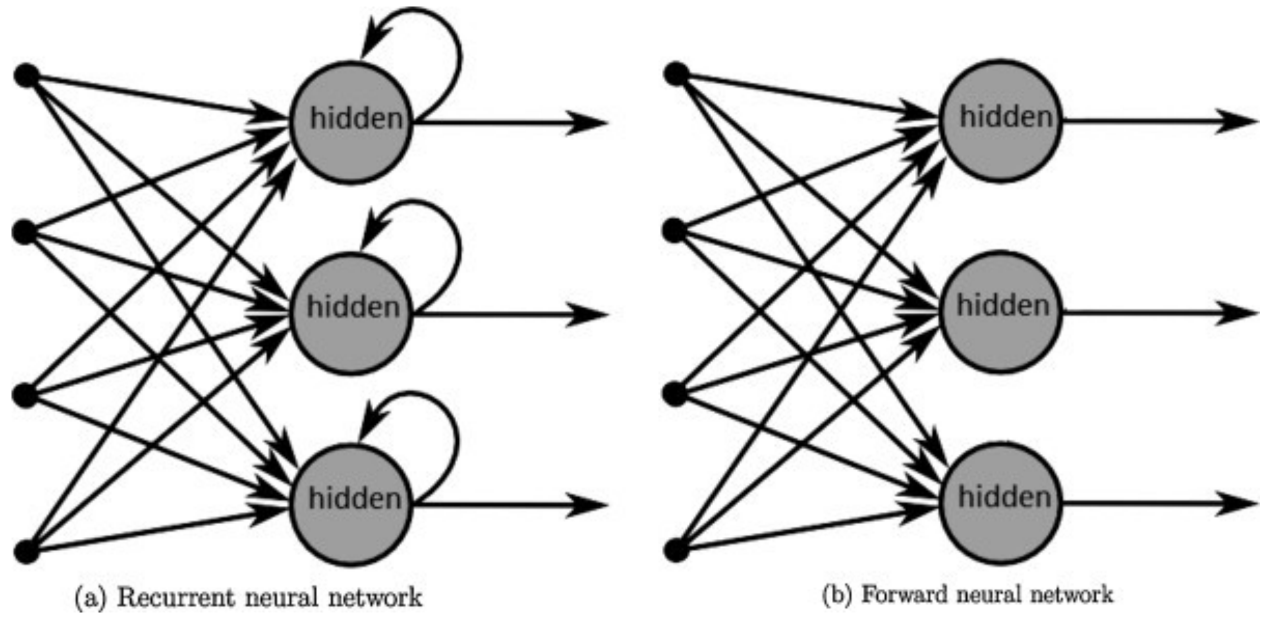


Fig. 6. Comparison of RNN and Forward neural network [24]

D. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a special kind of RNN, which has special forget gates, in addition to input and output gates of the simple RNN [18]. LSTMs are designed to remember the input state for a longer time than an RNN, hence allowing long sequences to be processed accurately. Wang et al in [25] presented an LSTM based model for POS tagging having an accuracy of 97%. LSTMs are a fundamental part of NLP architecture for Apple, Amazon, Google and other tech companies [16]. Fig 7 gives an overview of LSTM architecture.

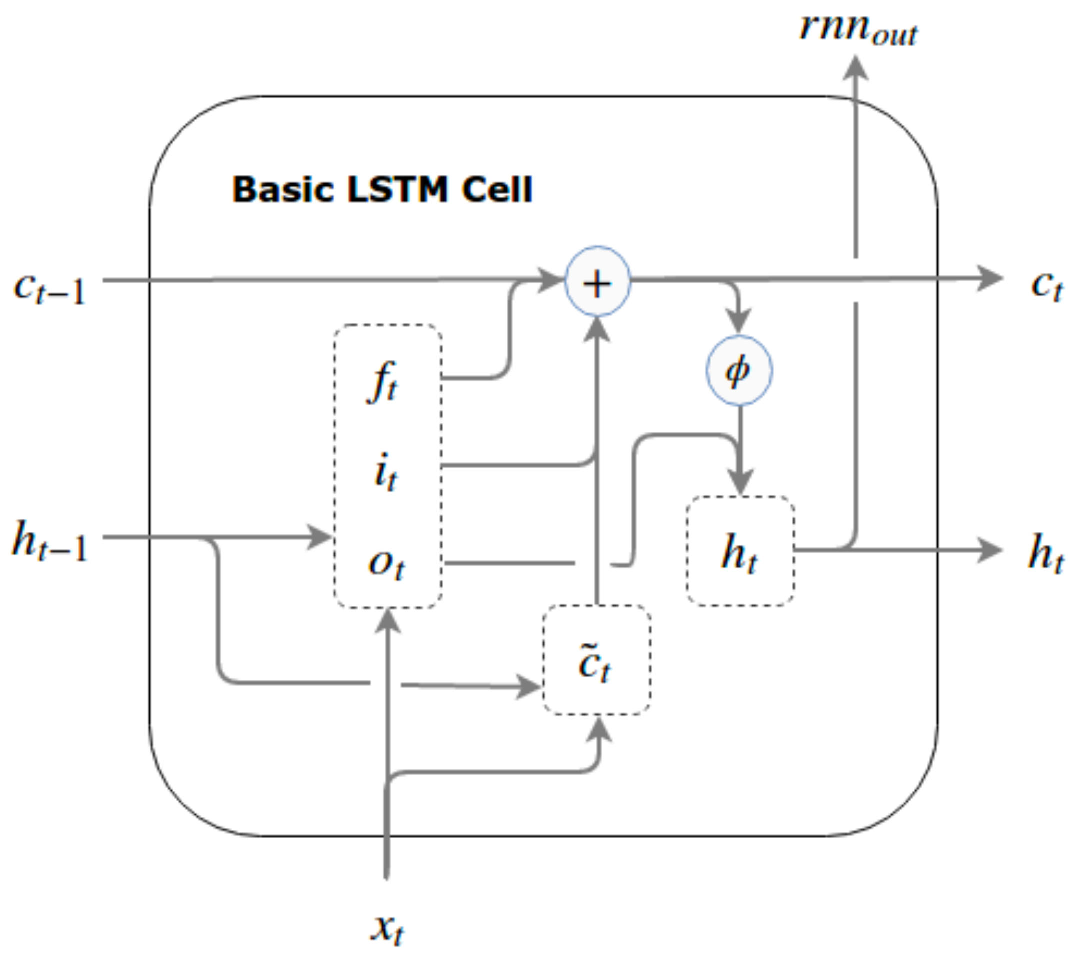


Fig. 7. An LSTM cell. The figure shows forget gate (f), input gate (i), output gate (o). [26]

E. Sequence to Sequence Models

Sequence to sequence models are based on RNN architecture and consists of two RNNs: an encoder and a decoder. The encoder's task is to process the input, and the decoder to process the output. Sequence to sequence models can be thought of as one decoder node producing output corresponding to one encoder node. This model has straightforward application in machine translation as a corresponding word for the output language can be generated by decoder easily by looking only at one word of input language at a time [27]. Fig. 8 shows a simple sequence to sequence model.

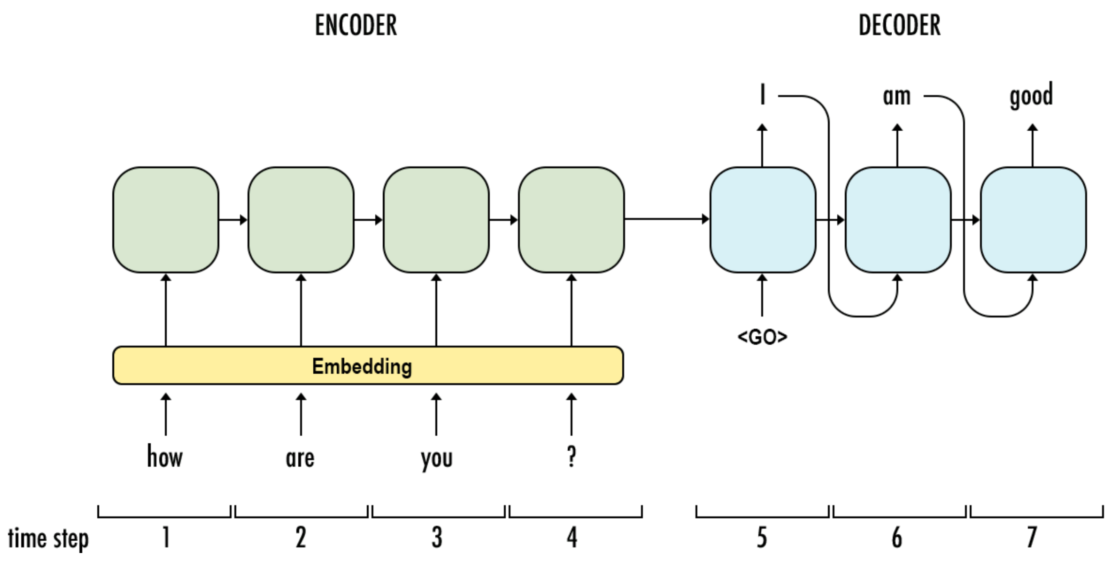


Fig. 8. Sequence to sequence model architecture [28]

V. NEURAL NETWORK BASED MODELS FOR CHATBOTS

A. Retrieval-based Neural Network

Retrieval-based models for chatbots have existed traditionally as rule-based answering systems. They have a fixed repository of responses mapped to questions [29]. Some sophisticated models, as used by Young et al. in [18] store context of the conversation, generating multiple responses based on the context, evaluating each response and outputting the response with the highest score. Retrieval-based systems are now coupled with Deep Learning techniques to provide more accurate responses. In their paper [30], Yan et al. have explored Deep Learning to analyze two sentences, hence having more context of the conversation and produce a response on a retrieval-based system. Such strategies provide much more accuracy as well as control over the chatbot. Fig. 9 shows a Retrieval based model architecture.

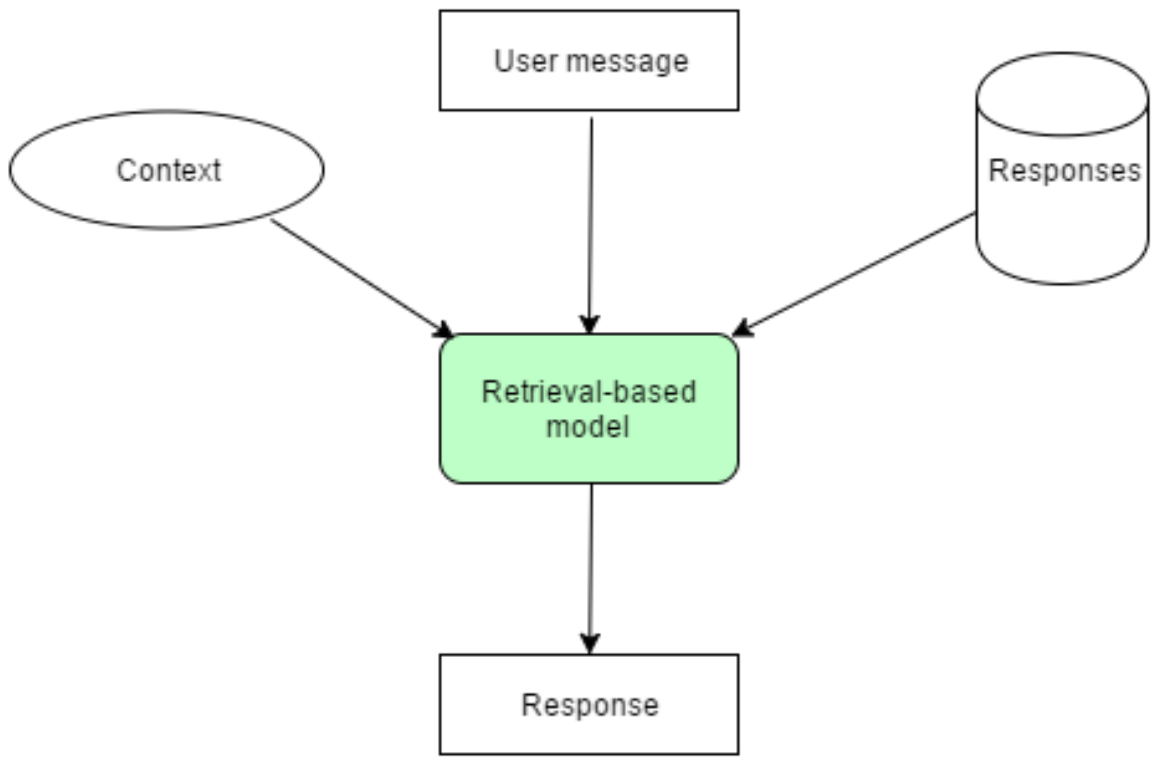


Fig. 9. Retrieval based model architecture [31]

B. Generation-based Neural Network

Generation-based Neural Networks, as opposed to Retrieval-based systems do not rely on fixed responses. They generate new responses from scratch [29]. Responses are based purely on machine learning and the training data. Sequence to sequence models, discussed in section III, are suited for Generation-based Neural Networks.

In [18], Young et al. used a Generation-based model in their chatbot for asking follow-up questions to the user, word by word. They used a template for their question (e.g. *"What about", "What do you think about"*) and generated the rest of the sequence by looking at the user input and the context. These models are currently being researched to work reasonably well and making them production-grade.

VI. SUMMARY OF THE CURRENT STATE-OF-ART

Deep Learning is a new, exciting domain with tremendous research being carried out in this space. This has allowed us to create generic models to analyze large data, at scale. Deep Learning helps eliminate the complexity of feature engineering from the traditional machine learning process and truly learns the underlying data patterns. Architectures like RNN, LSTM and sequence to sequence model overcome the limitation of context recognition, an essential part of NLP. Generation-based Networks have it made possible to create a “true” chatbot whose responses are based only on the training data.

These advances can be leveraged and explored to build life-like chatbots which can make real conversations with a human. By creating robust chatbots which can be customized according to the training data, large scale automation is possible.

Finally, the responses that a chatbot makes is mostly a “good guess”. NLP is a vast field, and for machines to truly understand the complexity of human interaction would require co-ordination of many branches of science such as psychology, literature and linguistics.

VII. HYPOTHESIS

A hybrid model for chatbots can be built, combining the control of a retrieval-based system, the context retention capabilities of LSTM, and the ease of sequence to sequence models. While Generative models are promising to build a life-like chatbot, they have a disadvantage in that they can make grammatical errors and can sometimes provide responses in inappropriate language, hindering their capability to be used in a production grade chatbot.

We will test the chatbot by making it interact with people, posing as a real person, and test how many lines of text it requires for a person to realize that it’s a chatbot. In addition to providing accurate responses, maintaining the context is also a challenge. We will test this by judging for how many lines of conversation the chatbot can retain context, and how well it performs after changing the context.

VIII. DATASET

The training data for a chatbot requires it to have a conversational flow. It needs to have a sentence or a question and a response. I found two data sources which fit this requirement: Cornell Movie Dialogs Corpus [32] and Reddit. I have chosen Reddit as my data source as it has a hierarchical structure and hence would require a little less processing. In the interest of storage space, I have chosen Reddit comments for January 2015 as my training set, which takes up approximately 45GB of storage.

The hierarchy of Reddit comments can be visualized as follows:

```
-Top level reply 1
--Reply to top level reply 1
--Reply to top level reply 1
---Reply to reply...
-Top level reply 2
--Reply to top level reply 1
-Top level reply 3
```

We want to transform this data into comment-reply pairs, so we can define input-output for the chatbot.

`-Top level reply 1` and `--Reply to top level reply 1`

`--Reply to top level reply 1 and ---Reply to reply...`

An example of a comment-reply pair can be:

Comment: Spent my childhood watching my 16 years older cousin play videogames. Love to watch streams as an adult.

Reply: Yeah similar for me. I watch Super Beard Bros all the time now because its exactly like sitting around and cracking jokes with friends. Or I'll watch Cryaotic if I want that super laid back story driven game.

The downloaded data has a format as follows:

```
{"author":"Arve","link_id":"t3_5yba3","score":0,"body":"Can we
please deprecate the word \"Ajax\" now? \r\n\r\n(But yeah, this
_is_ much
nicer)","score_hidden":false,"author_flair_text":null,"gilded":0,"s
ubreddit":"reddit.com","edited":false,"author_flair_css_class":null
,"retrieved_on":1427426409,"name":"t1_c0299ap","created_utc":"11924
50643","parent_id":"t1_c02999p","controversiality":0,"ups":0,"disti
nguished":null,"id":"c0299ap","subreddit_id":"t5_6","downs":0,"arch
ived":true}
```

We do not need all of these attributes. Our attributes of interest are “body”, “comment_id”, “parent_id”, “subreddit” and “score”.

IX. PREPROCESSING

A. SQLite

The size of this data is very high (45GB). Hence, it is necessary to buffer through this data. Additionally, we had to remove the unnecessary data which would also reduce the size of the files. To solve both of these problems, we decided to select the desired data and store it in an SQLite database.

The insertion operation on the database was conducted through a python 3 script, using SQLite3 libraries. The script took ~8.5 hours to run on MacBook Pro 2.3 GHz Intel Core i5 16 GB RAM.

B. Word Embedding

The words in our training data can be represented as one-hot vectors. This is done by capturing a corpus of all the English words (~10,000). Each word in the input is then represented as a vector of 10,000 values, with all zeros except the subscript of the word from our corpus. For example, if Orange is the 6000th word in the corpus, the one hot vector for Orange is represented as a vector of 10,000 values, all zeros except 6000th index in the vector as 1.

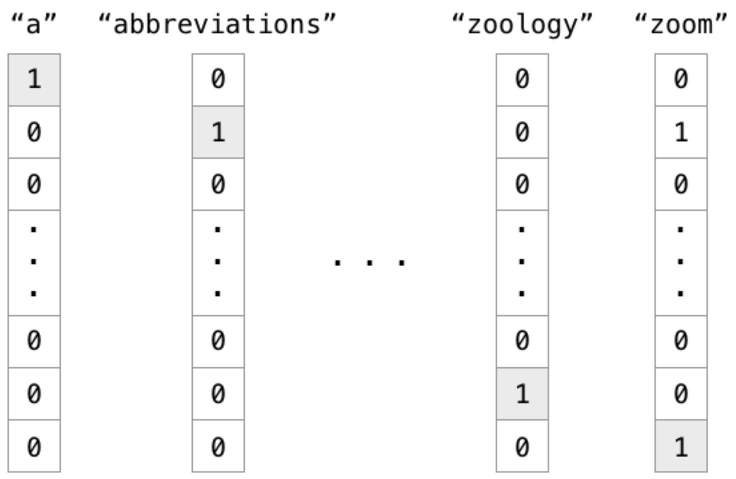
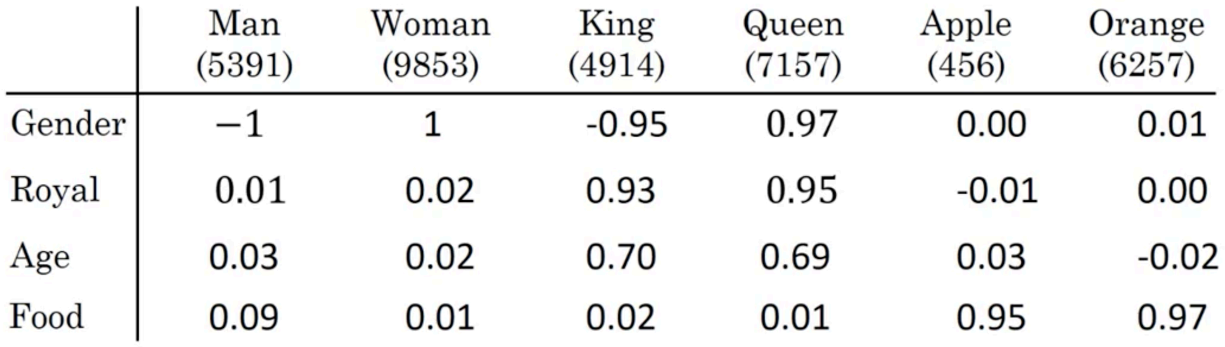


Fig. 10. One-hot representation of words [33]

One hot vectors are widely used for their advantages in increasing the overall performance of the model. However, it is not sufficient when it comes to NLP tasks. This is because our input will be a sequence of words. If we simply calculate the one hot vectors for each input word, we are not able to capture the relationship between them, which is the essence for our chatbot.

Word embedding is a technique to capture the relationship between words. We first decide the features we want to capture from the words. In the example in Fig. 11, the column headers represent the words and row headers represent the features. Each corresponding value represents the association of that word with each feature. The value in parentheses for each word is its number in the corpus of words.



	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

Fig. 11. Word Embedding Matrix Example

Take the word "Man" as an example. The column under "Man" represents the value corresponding to the feature. Since "Man" is closely associated with Gender, it has a strong negative value. Similarly, "Woman" has a strong positive value. This trend can be noticed in "King" and "Queen" as well. Since "Orange" and "Apple" have no association with gender, they have values close to 0.

The embedding is usually represented as a matrix. To get the word embedding for a particular word, we just multiply the embedding matrix with the one-hot vector of the word. This reduces storage complexity of the application while providing way to capture the relationship between words.

Learning word vector representations is an unsupervised learning problem. There are several ways to do this effectively, the latest being Global Vectors for Word Representation

(GloVe) [34]. It was developed by J. Pennington *et al.* as part of Stanford NLP projects. They have provided a pre-trained model for GloVe representation to download. We have used this data and conducted transfer learning for our chatbot.

X. CAPTURING CONTEXT

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

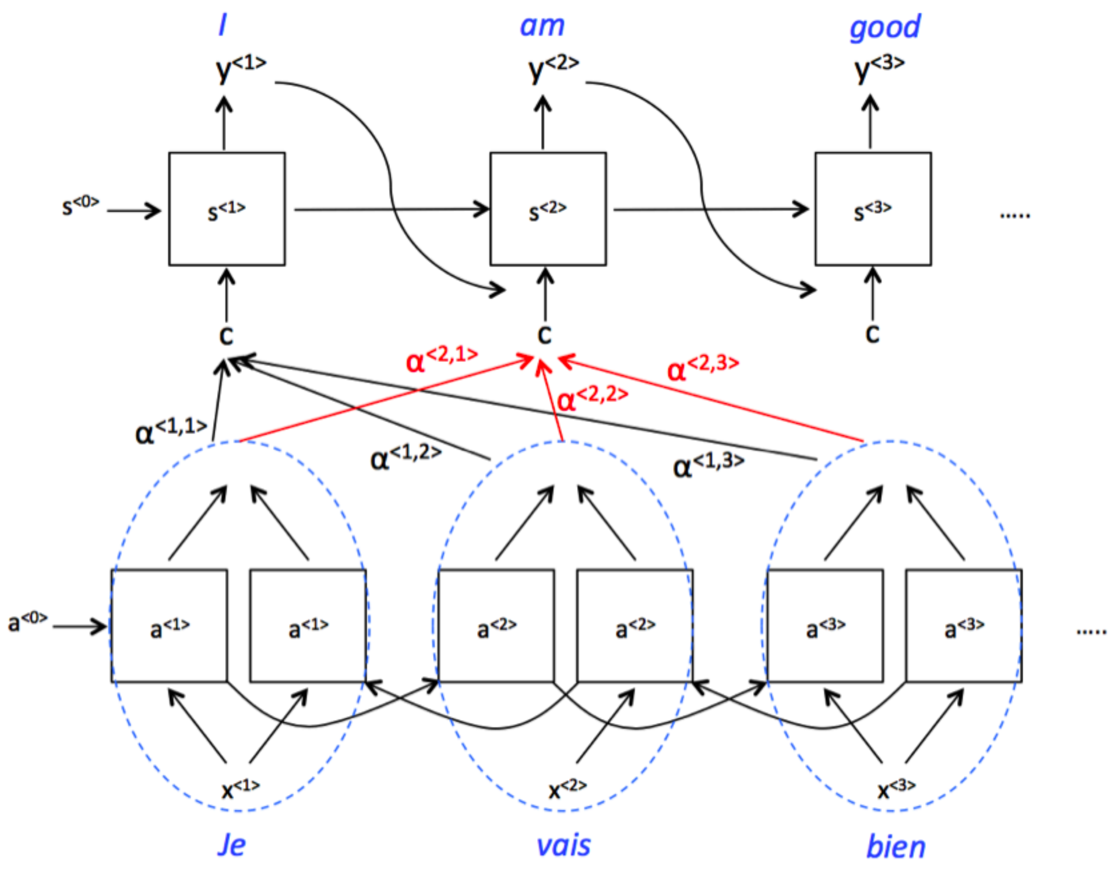
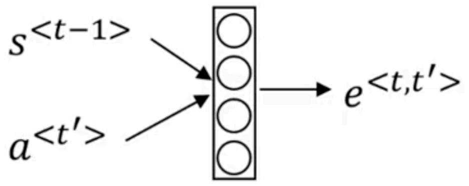


Fig. 12. Attention model for capturing context proposed for Machine Translation [35]

Formally, we can represent this as a parameter α<t, t′> which represents the attention given to t′ token to generate t. It can be visualized in a network as shown in Fig. 11. α<1,2> represents the contribution of "vais" for generating the English translation of "Je". The values for α can be calculated by the formula given below.

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

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



From Fig. 12 we can infer that the attention weights will transfer the attention from the encoder to the decoder for machine translation. Since the output of the decoder is fed as input to the next time step, we consider s<t-1> and a<t′>, the output of the encoder at time t′ to calculate e<t, t′>. Since this relation is difficult to determine and can be different for each case, we will use gradient descent to learn the representation. In this way, we have included context in our model.

XI. METRICS TO TRACK

A. BLEU Score

The BLEU (BiLingual Evaluation Understudy) [36] score, proposed by Papineni *et al.* is a method for comparing a generated sequence of words to a reference sequence. This score was initially developed for translations but can be generalized to all NLP tasks.

The advantages of BLEU are as below:

- Fast and inexpensive calculation
- Easy to interpret
- Closest to human evaluation
- Standard metric for NLP tasks

The basic idea is to compare the number of times a word (or n-gram) appears in the machine output to the reference output, then average over it.

BLEU Score is calculated with the below formula:

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

Where BP is the brevity penalty which penalizes short sequences, allowing longer sequences to be scored better.

For our chatbot, we have the reference response as defined in our database. We will use the “reply” from comment-reply pairs from the database as the reference string. For each new query, our chatbot will generate a response. We will then calculate the BLEU scores for each response. Finally, after evaluating 50 responses for each model, we will average them and consider an aggregated score.

There are a few drawbacks of this metric. It does not consider the sequence of words, just the number of times the word appears in the machine generated sequence. Hence, BLEU scores are only close to human evaluation, but cannot match human evaluation. This led us to conduct the Turing Test in addition to BLEU Scores.

B. Turing Test

Turing Test, as explained in Section III, is a test to check how indistinguishable a chatbot is from a real person. We are exploring this test because BLEU scores, even though they are standard, cannot measure the quality of our responses as a human can do. With the Turing Test, we are trying to measure whether a chatbot's responses seem real.

To conduct the Turing Test, we have chosen 15 people who do not know that they are chatting with a chatbot. After having a conversation with it for more than 20 utterances, they were asked a question: "On a scale of 1 to 10, how natural was the conversation?". These responses are then averaged and normalized to get an aggregated score. The people selected come from various backgrounds, half of them from engineering background and half non-technical. This makes the test more robust.

XII. RETREIVAL-BASED CHATBOT PERFORMANCE

To test the performance of a retrieval based chatbot, we have used the Alicebot which is hosted on Pandorabots.com. As mentioned in Section III, Alicebot is one of the best retrieval-based chatbots, winning the Loebner prize three times. As discussed earlier, we will conduct the BLEU score test and Turing Test.

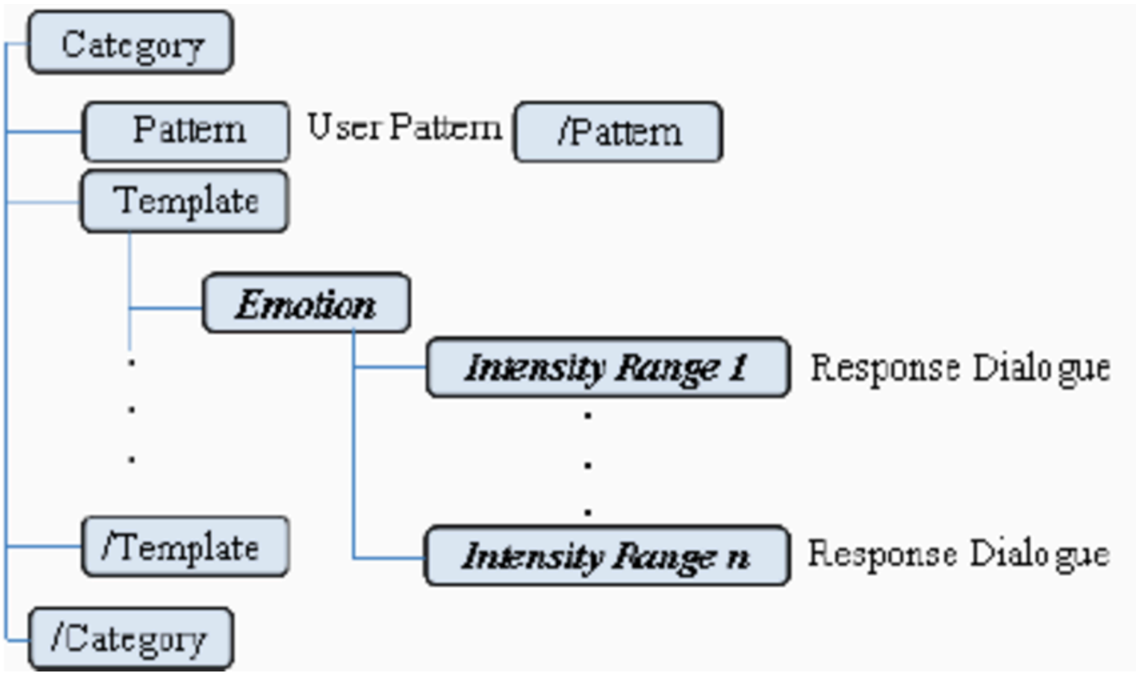


Fig. 13. AIML Language Extension [37]

Fig. 13 shows the general architecture of AIML. There are simple tags for pattern matching, and responses are predefined. The developer has to decide the data flow and write logic for selecting the appropriate response. The structure can be as complex as the developer wants it to be. We have performed experiments on the standard Alicebot.

Sample Conversation:

User: Hi

Alice: Hello. My name is Alice and I'm VERY pleased to meet you.

User: How are you today?

Alice: Ah. I am very well thank you for asking. very well thank you for asking.

User: Where is San Jose?

Alice: it is a city in California.

User: Which part of California?

Alice: The first one, I think.

User: What do you mean?

Alice: Maybe I didn't understand your meaning.

A. Results

1. BLEU Score: 0.92
2. Normalized Turing Test Score: 0.69

B. Analysis

- The BLEU score of this chatbot is close to 1, the maximum score. This indicates that it has given more meaningful responses, close to what a human would give. This is expected as the responses are predefined, by a human.
- The Turing Test score is 17% less than the BLEU score. This indicates that people did not find the responses convincing enough to see the program as a human. This can be observed from the sample text above. Alice is unable to answer follow up questions, hence, cannot effectively hide that it is a program.

XIII. GENERATION-BASED CHATBOT ARCHITECTURES

This part of the project required us to conduct experiments on different architectural designs. We have followed certain guidelines to construct the NLP models. However, Deep Learning is still a new paradigm of AI, and the optimum results can only be obtained by conducting many experiments by changing hyper parameters of the network.

A. Unidirectional LSTM

To start off, we will consider a unidirectional LSTM layer. We will see how the data will flow and change hyper parameters like the number of neurons and regularization.

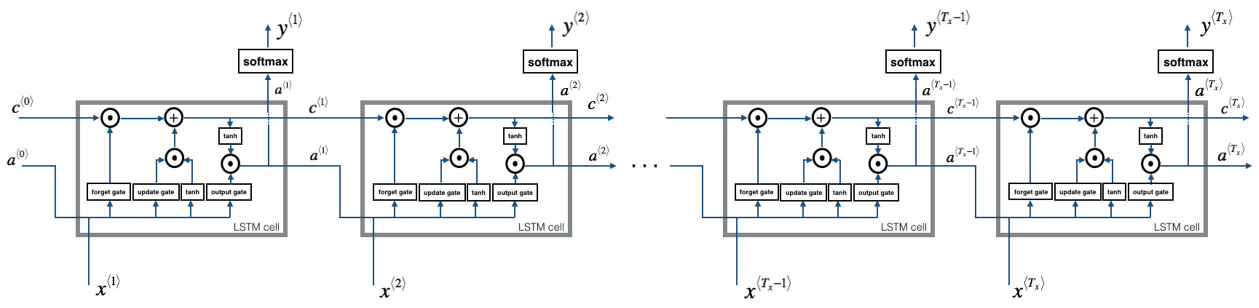


Fig. 14. Unidirectional LSTM Architecture [38]

a<0> and c<0> represents the hidden state and state of memory cell at time step 0, respectively. x<1> is the input at the first timestep, which is the first input word in our case. In LSTMs, the hidden state is propagated to the next time step, which helps it to retain context. In case of LSTM, the more timesteps we look at significantly adds to the complexity of the model, ultimately increasing the runtime for training. For simplicity, we are looking at 10 words at a time, and generating 10 words. To indicate end of sentence, we have added <eos> tag to the end of the input sequence. Any input shorter than 10 words is padded with <unk> tag. Any input longer than 10 words will be considered as overflow.

Sample Conversation:

User: Hi

Bot: Hi

User: How are you doing today?

Bot: I am I am good, how are you?

a) Results (For 600 neurons case)

1. BLEU Score: 0.65

2. Normalized Turing Test Score: 0.63
3. Train time: 4 hours 29 minutes

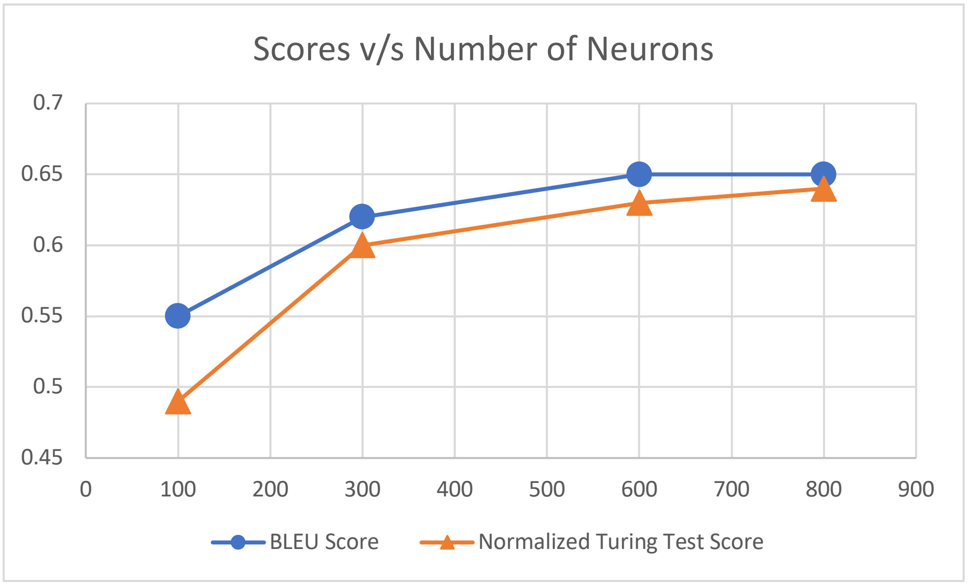
b) Analysis

- The BLEU score of this model is 21% lower than retrieval-based chatbot. This indicates that the chatbot is incapable of generating meaningful responses. This can be observed from the sample conversation, where the model is not able to generate a simple sentence.
- The Turing Test score follows the trend of BLEU score. This is expected, as the BLEU scores were low as well.

We tried to tune our model by changing the number of neurons.

Number of Neurons	**BLEU Score**	**Normalized Turing Test Score**
100	0.55	0.49
300	0.62	0.6
600	0.65	0.63
800	0.65	0.64

The graph of these results can be plotted as:



As can be seen from the results, the values are not changing much even with adding more neurons. Since the scores are minimally improving for 800 neurons case as opposed to 600 neurons case, we have chosen 600 neurons to save training time. The model is unable to produce a single sentence, which clearly indicates that it requires more complexity. Also, since text data is sequential, the unidirectional LSTM might not suffice for this task. We will try bidirectional LSTM next.

B. Bidirectional LSTM

The bidirectional LSTM takes inputs from previous time step as well as the next time step. In our case, this allows the LSTM to look at surrounding words to generate a new sequence. We expect this architecture to perform better than Unidirectional Architecture.

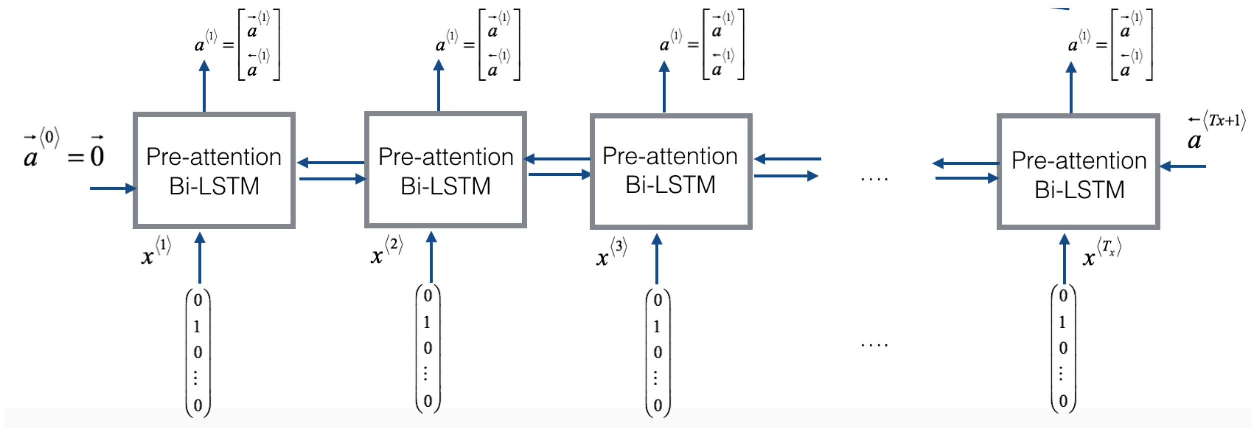

$a^{\langle 1 \rangle} = \begin{bmatrix} \overrightarrow{a}^{\langle 1 \rangle} \\ \overleftarrow{a}^{\langle 1 \rangle} \end{bmatrix}$
$\overrightarrow{a}^{\langle 0 \rangle} = \vec{0}$
Pre-attention Bi-LSTM
$x^{\langle 1 \rangle}$
$x^{\langle 2 \rangle}$
$x^{\langle 3 \rangle}$
$x^{\langle T_x \rangle}$
$\overleftarrow{a}^{\langle Tx+1 \rangle}$
....

Fig. 16. Bidirectional LSTM Architecture [38]

Fig. 16 shows Bidirectional LSTM architecture. It is same as unidirectional, except that there are two values of *a* which are generated for each time step. We concatenate these values to get the final output of that time step.

Sample Conversation:

User: Hi

Bot: Hi

User: How are you doing today?

Bot: I am good. How are you?

User: Do you know who is Kanye West?

Bot: No

User: He’s great

Bot: I’m not sure who is

User: You’re not sure who is what?

Bot: Yes

a) Results (for 600 neurons case)

1. BLEU Score: 0.69
2. Normalized Turing Test Score: 0.68
3. Training time: 5 hours 38 minutes

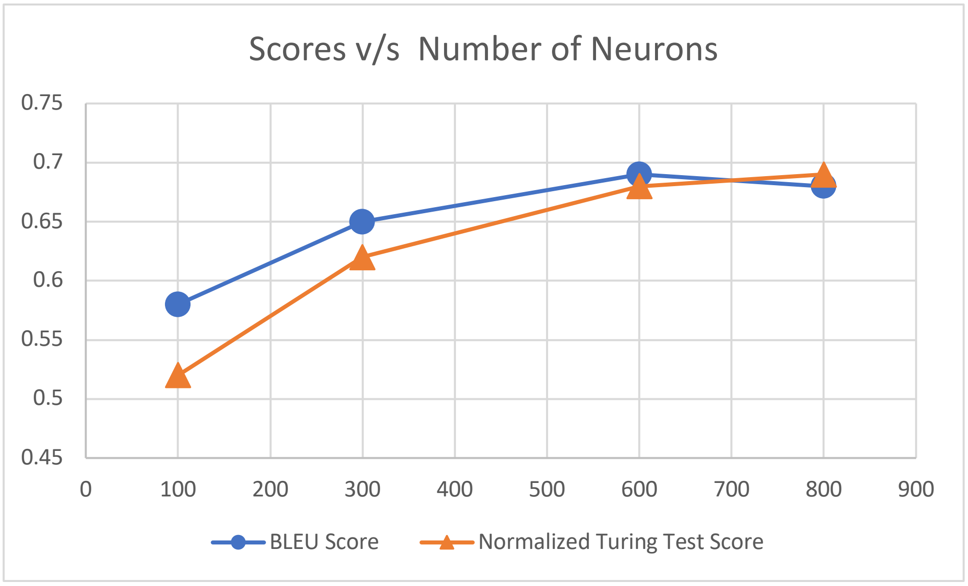
b) Analysis

- The BLEU score of this model is 3% higher than the Unidirectional LSTM. This is expected.
- The Turing Test score is 7% higher than Unidirectional LSTM.

We tried to tune our model by changing the number of neurons.

Number of Neurons	**BLEU Score**	**Normalized Turing Test Score**
100	0.58	0.52
300	0.65	0.62
600	0.69	0.68
800	0.68	0.69

The graph of these results can be plotted as:



The trend is similar to Unidirectional LSTM. After a certain point, the scores are not changing much. It can be observed from the sample conversation that the model cannot keep context of the sentence. This indicates that we need to incorporate the attention mechanism in our chatbot.

C. Unidirectional LSTM with attention mechanism

Next, we have conducted experiments by incorporating attention mechanism into the model.

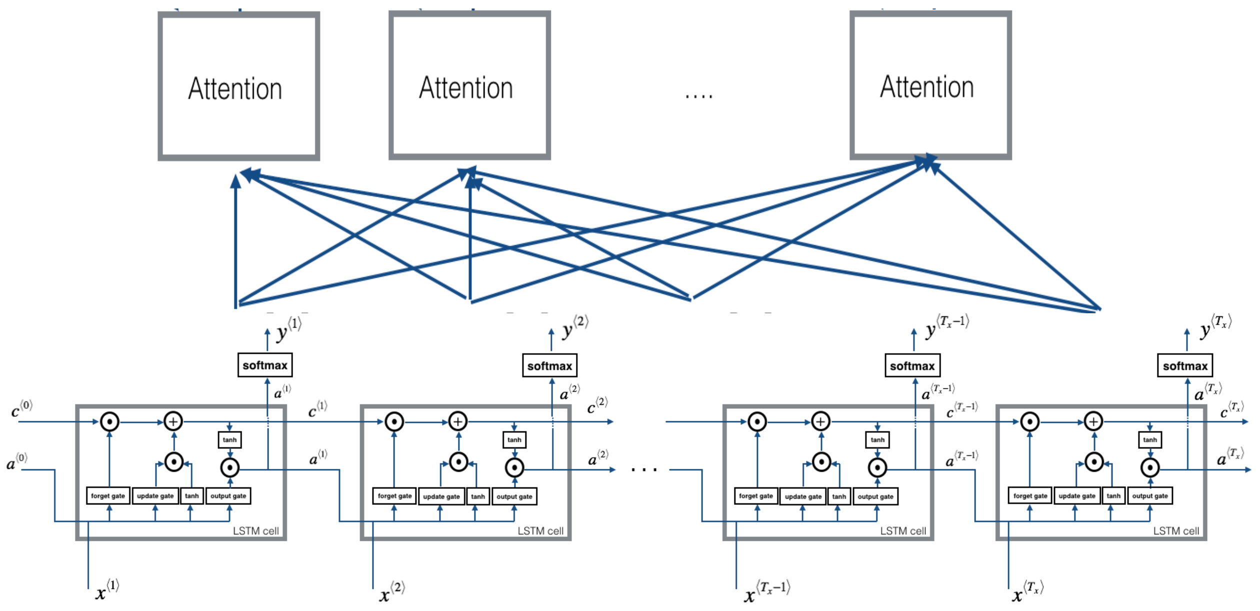


Fig. 17. Unidirectional LSTM with Attention [38]

Fig. 17 shows Unidirectional LSTM with attention. The attention mechanism is implemented as discussed in Section X.

Sample Conversation:

User: Hello!

Bot: Hey

User: How are you?

Bot: I’m doing great

User: Do you know who is Kanye West?

Bot: He is he is a great

a) Results

1. BLEU Score: 0.73

2. Normalized Turing Test Score: 0.72
3. Training Time: 10 hours 44 minutes

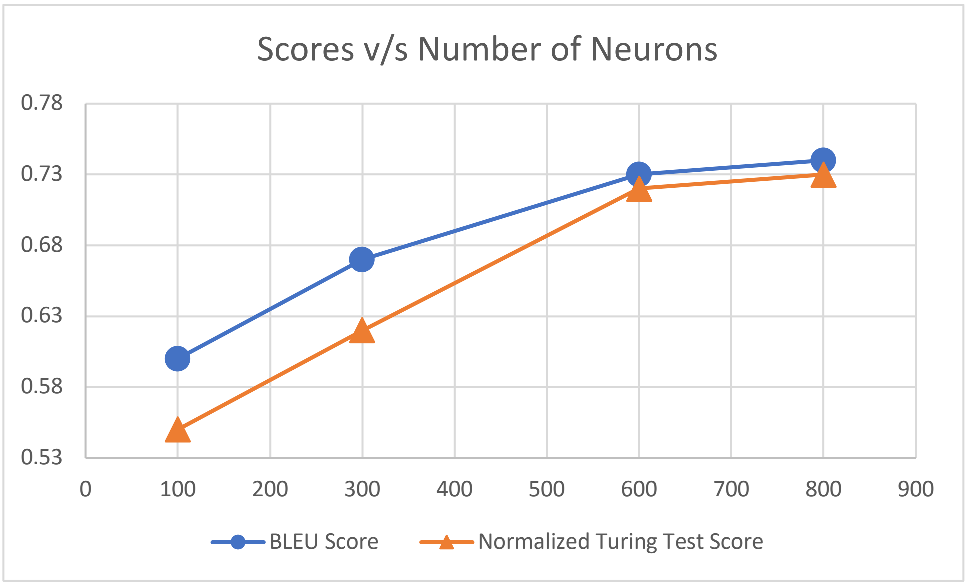
b) Analysis

- The BLEU score of this model is a 6% higher than the Bidirectional LSTM. This is expected, as the attention mechanism allows us to pay attention to surrounding words while generating output. This has helped the model to capture context, which was Kanye West in this case.
- The Turing Test score is a 5.8% higher than Bidirectional LSTM. This follows the increase in BLEU score. The model, however, shows the same drawbacks of the Unidirectional LSTM. It struggles with generating meaningful sentences. This cannot be used as the best model.

We tried to tune our model by changing the number of neurons.

Number of Neurons	**BLEU Score**	**Normalized Turing Test Score**
100	0.60	0.55
300	0.67	0.62
600	0.73	0.72
800	0.74	0.73

The graph of these results can be plotted as:



Since the results for 600 neurons case and 800 neurons case are not very different, we will use 600 neurons because it takes less time to train.

D. *Bidirectional LSTM with Attention*

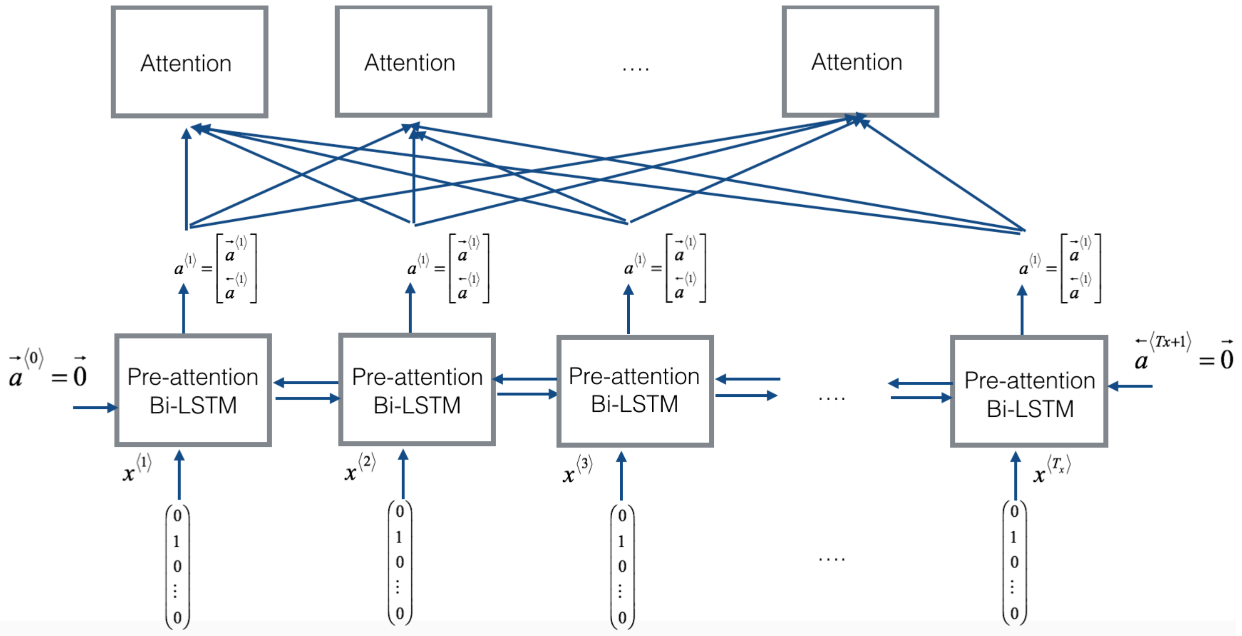

Attention
Attention
....
Attention
$a^{\langle 1 \rangle} = \begin{bmatrix} \vec{a}^{\langle 1 \rangle} \\ \overleftarrow{a}^{\langle 1 \rangle} \end{bmatrix}$
$\vec{a}^{\langle 0 \rangle} = \vec{0}$
Pre-attention Bi-LSTM
$\overleftarrow{a}^{\langle Tx+1 \rangle} = \vec{0}$
$x^{\langle 1 \rangle}$
$x^{\langle 2 \rangle}$
$x^{\langle 3 \rangle}$
$x^{\langle T_x \rangle}$
....

Fig. 18. Bidirectional LSTM with Attention [38]

Fig. 18 shows Bidirectional LSTM with attention. The attention mechanism is implemented as discussed in Section X.

Sample Conversation:

User: What’s up?

Bot: Nothing much

User: Tesla is launching a new truck

Bot: Really, bankrupt

a) Results (for 600 neurons case)

1. BLEU Score: 0.76
2. Normalized Turing Test Score: 0.72
3. Training Time: 11 hours 58 minutes

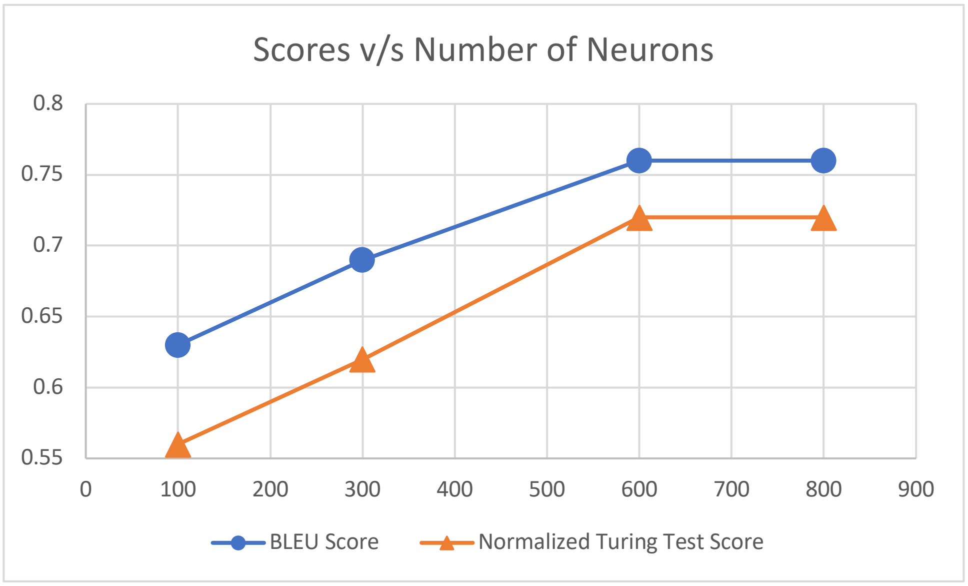
b) Analysis

- The BLEU score of this model is 4% higher than the Bidirectional LSTM. This is expected, as these features combine two powerful mechanisms.
- The Turing Test Score has not changed. This might be the case as the model is still struggling to output meaningful sentences, even after capturing the context. This clearly shows that there must be an LSTM layer at the output.

We tried to tune our model by changing the number of neurons.

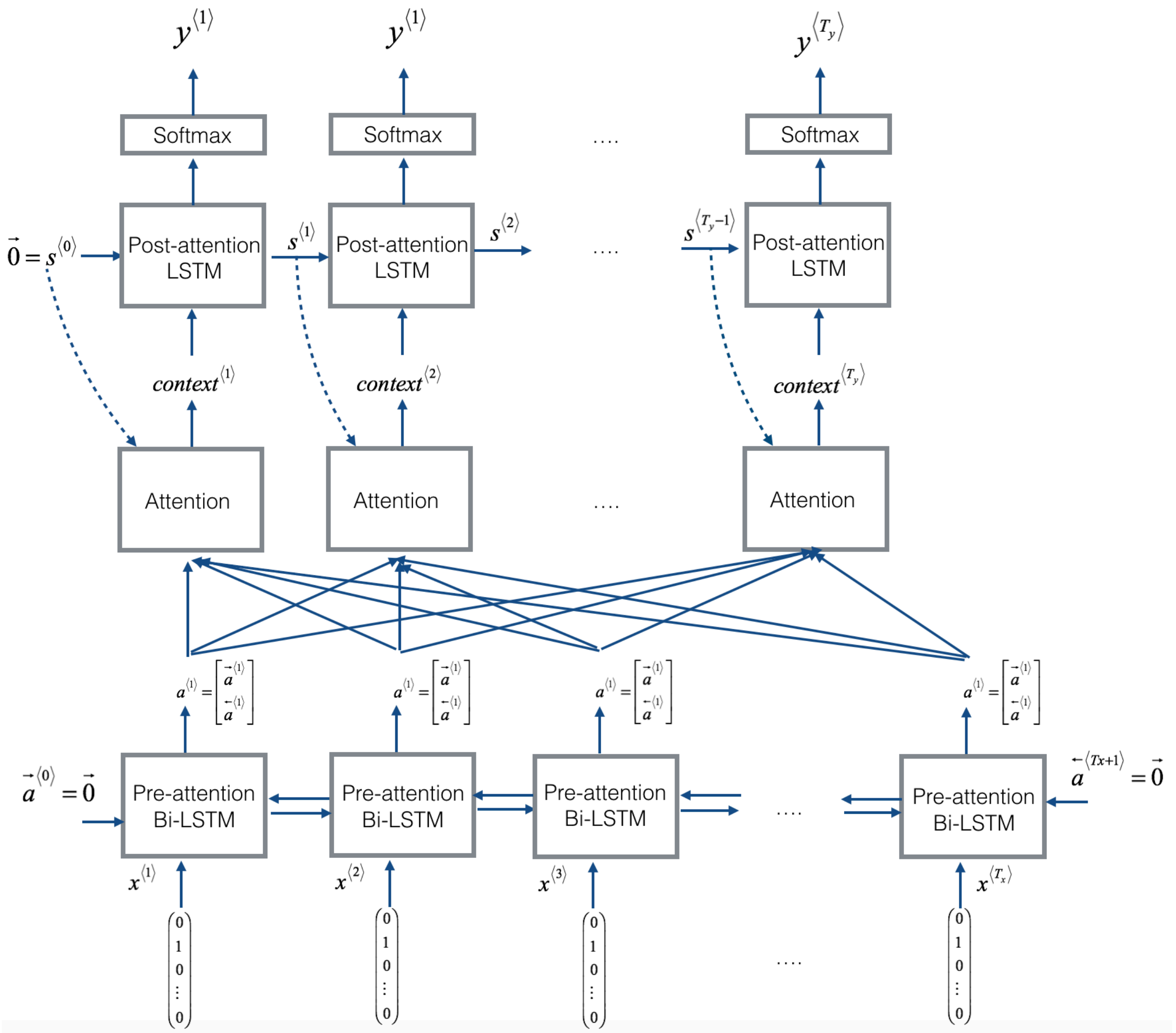
Number of Neurons	**BLEU Score**	**Normalized Turing Test Score**
100	0.63	0.56
300	0.69	0.62
600	0.76	0.72
800	0.76	0.72

The graph of these results can be plotted as:



Similar to the other cases, increasing the number of neurons more than 600 is not significantly increasing our scores.

E. Bidirectional LSTM with Attention, with additional LSTM layer



$y^{\langle 1 \rangle}$
$y^{\langle 1 \rangle}$
$y^{\langle T_y \rangle}$
Softmax
Softmax
Softmax
$\vec{0} = s^{\langle 0 \rangle}$
Post-attention LSTM
$s^{\langle 1 \rangle}$
$s^{\langle 2 \rangle}$
$s^{\langle T_y - 1 \rangle}$
$context^{\langle 1 \rangle}$
$context^{\langle 2 \rangle}$
$context^{\langle T_y \rangle}$
Attention
$a^{\langle 1 \rangle} = \begin{bmatrix} \vec{a}^{\langle 1 \rangle} \\ \overleftarrow{a}^{\langle 1 \rangle} \end{bmatrix}$
$\vec{a}^{\langle 0 \rangle} = \vec{0}$
Pre-attention Bi-LSTM
$\overleftarrow{a}^{\langle Tx+1 \rangle} = \vec{0}$
$x^{\langle 1 \rangle}$
$x^{\langle 2 \rangle}$
$x^{\langle 3 \rangle}$
$x^{\langle T_x \rangle}$

Fig. 19. Bidirectional LSTM with attention and additional LSTM layer [38]

Fig. 19 shows Bidirectional LSTM with attention and additional LSTM layer. The attention mechanism is implemented as discussed in Section X.

Sample Conversation:

User: Hey!

Bot: Hey, how are you?

User: I’m good, how are you?

Bot: I’m good too

User: Where do you live?

Bot: I live in Baltimore

User: Do you know Arjun from Mahabharata?

Bot: Arjun, you could not do this.

a) ***Results (for 600 neurons case)***

1. BLEU Score: 0.79
2. Normalized Turing Test Score: 0.78
3. Training Time: 14 hours

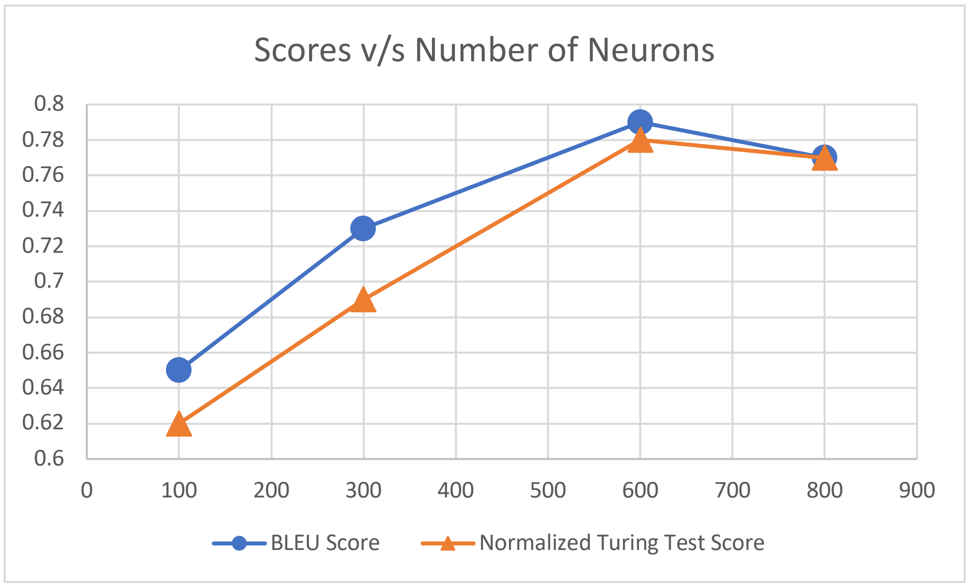
b) ***Analysis***

- The BLEU score of this model is 4% higher than all the other models.
- The Turing Test Score has improved significantly, 8% more than the previous model.

We tried to tune our model by changing the number of neurons.

Number of Neurons	**BLEU Score**	**Normalized Turing Test Score**
100	0.65	0.62
300	0.73	0.69
600	0.79	0.78
800	0.77	0.77

The graph of these results can be plotted as:



Although this model is the best chatbot yet, it has some drawbacks. Whenever it is asked something that it has no knowledge about, it outputs something random. This is the general drawback of the generative model based chatbot. If asked about something that it does not have knowledge about, we expect the chatbot to answer that it doesn't know about it. This can be ensured by using a hybrid model, combining the rule based and generative based approaches. We will next combine this model and Alicebot, discussed in part A of this section.

F. Combining Generative and Retrieval-based Model

We have combined the Alicebot model and the best generative based model for this experiment. This experiment is expected to have highest BLEU score and Turing Test score from all the discussed models.

Sample Conversation:

User: Hello, how are you?

Bot: I'm fine, how are you?

User: I'm great!

User: Do you go to school?

Bot: I do but I don’t attend classes that much

User: Do you know Arjun from Mahabharata?

Bot: Sorry, I don’t know about that

a) Results

1. BLEU Score: 0.85
2. Normalized Turing Test Score: 0.89
3. Training Time: 14 hours

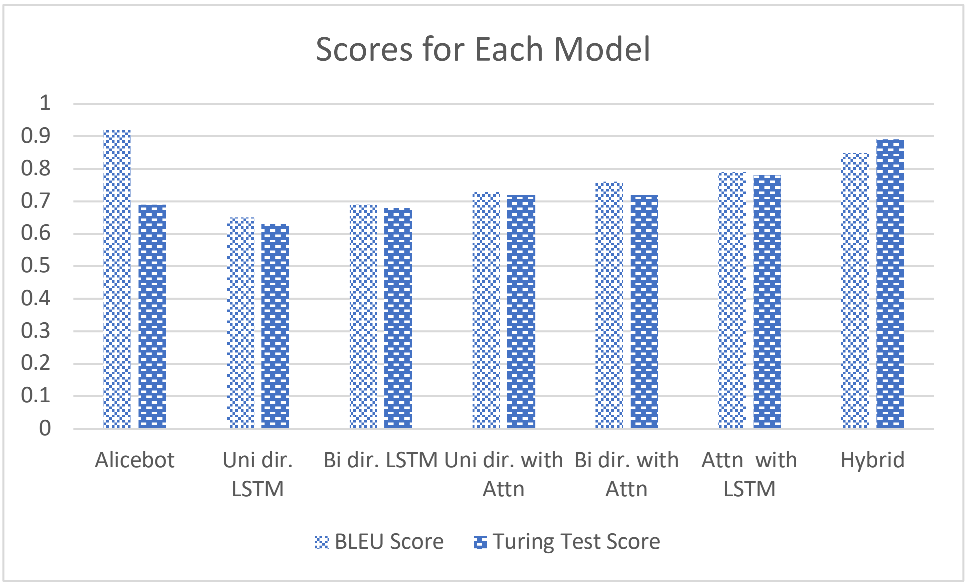
b) Analysis

- The BLEU score of this model is the highest among all the generation-based models.
- The Turing Test Score has improved by 28% as compared to Alicebot.

The sample conversation shows that the chatbot can now generate meaningful sentences, without losing context and have a natural conversation.

XIV. CONCLUSION

The graph below summarizes the results for BLEU scores and Turing Test scores of all models:



The first model, which was Alicebot, has the highest BLEU score (0.92). This is expected as the responses were hard-coded by a human. Alicebot's score for Turing test is 17% lower than its BLEU score. This is because it is simply an if-else block, with hardcoded responses. Alicebot is not a truly feasible solution because it is nearly impossible for a human to encode all the responses in advance. As we started exploring generation-based models with Unidirectional LSTM, it can be seen that the BLEU scores and Turing test scores gradually increase, before achieving the best results with the hybrid model. We can conclude that combining generation-based model with rule-based model helps us achieve the best combination of BLEU score and Turing test score.

Out of all the experiments conducted, the hybrid model has the best overall scores. Alicebot still has the best BLEU score, but the hybrid model's BLEU score is only 7% less than that of Alicebot. Since the hybrid model generates each response from scratch, it's BLEU score can be considered significant.

XV. FUTURE WORK

This chatbot is trained on static data, from January 2015. Hence, the data doesn't capture the latest information available, like current events. Ideally, we would want the model to be trained automatically after a brief period of time. The training time of the model, which is 14 hours, is a hindrance in this case. Some experimentations can be done pertaining to reducing the training time. There are some new publications from 2018 [39] which have combined CNN and LSTM. The advantage of this architecture is that the model complexity is reduced, making the training time faster.

Deep Reinforcement Learning (DRL) is a complex Deep Learning technique which involves having an agent interact with an environment [40]. Unlike the traditional way of training a model, DRL algorithms reward the agent when desired results are produced. This reinforces positive behavior in the agent. A few experts see DRL as a path to Artificial General Intelligence [40]. Recent developments and growing confidence in DRL space have led to releasing new open source toolkits like OpenAI Gym to make DRL easily accessible to researchers and developers. A major advantage of DRL with respect to chatbots is we can judge and reward a continuous dialogue with multiple sentences, as opposed to a single sentence at a time [28]. This makes exploring DRL extremely lucrative to be used for chatbots.

REFERENCES

[1] M. Lewkowitz, "Bots: The future of human-computer interaction.," 12 Feb 2014. [Online]. Available: https://chatbotsmagazine.com/bots-the-future-of-human-computer-interaction-56696f7aff56.

[2] J. Vanian, "Google Adds More Brainpower to Artificial Intelligence Research Unit in Canada," Fortune, 21 November 2016. [Online]. Available: http://fortune.com/2016/11/21/google-canada-artificial-intelligence/.

[3] I. Goodfellow, Deep learning, Cambrigde, MA: The MIT Press, 2016.

[4] B. Copeland, "MYCIN," in *Encyclopædia Britannica, Inc.*, 2017.

[5] Y. Lecun, Y. Bengio and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, p. 436, 2015.

[6] Neural Networks and Deep Learning, "Why are deep neural networks hard to train?," [Online]. Available: http://neuralnetworksanddeeplearning.com/chap5.html.

[7] M. L. Mauldin, "ChatterBots, TinyMuds, and the Turing test: entering the Loebner Prize competition," in *Proceedings of the 12th National Con- ference on Artificial Intelligence*, 1994.

[8] J. Weizenbaum, "ELIZA - A Computer Program for the Study of Natural Language Communication Between Man and Machine," *Communications of the ACM,* vol. 26, no. 1, pp. 23-28, Jan 1983.

[9] Y. Vilner, "Chatbots 101: The Evolution of Customer Retention's Latest Trend," 20 July` 2017. [Online]. Available: https://www.entrepreneur.com/article/293439.

[10] M. J. Pereira, L. Coheur, P. Fialho and R. Ribeiro, "Chatbots' Greetings to Human-Computer Communication," 2016.

[11] R. Raine, "Making a clever intelligent agent: The theory behind the implementation," in *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, 2009.

[12] alicebot.org, "Alicebot Technology History," [Online]. Available: http://www.alicebot.org/history/technology.html.

[13] Futurism, "The History of Chatbots," [Online]. Available: https://futurism.com/images/the-history-of-chatbots-infographic/.

[14] Chatbots.org, "Smarterchild," [Online]. Available: https://www.chatbots.org/chatterbot/smarterchild/.

[15] I. N. d. Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni and S. F. d. R. Alves, Artificial Neural Networks A Practical Course, Springer International Publishing, 2017.

[16] O. Davydova, "7 Types of Artificial Neural Networks for Natural Language Processing," [Online]. Available: https://www.kdnuggets.com/2017/10/7-types-artificial-neural-networks-natural-language-processing.html.

[17] G. M and D. S. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1352231097004470.

[18] T. Young, D. Hazarika, S. Poria and E. Cambria, "Recent Trends in Deep Learning Based Natural Language Processing".

[19] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *Proceedings of the 25th international conference on machine learning*, 2008.

[20] D. Duncan, "Quora," [Online]. Available: https://www.quora.com/How-common-is-it-for-neural-networks-to-be-represented-by-3rd-order-tensors-or-greater.

[21] T. Arda, H. Véronique and M. Lieve, "A Neural Network Architecture for Detecting Grammatical Errors in Statistical Machine Translation," *Prague Bulletin of Mathematical Linguistics,* vol. 108, no. 1, pp. 133-145, 1 June 2017.

[22] A. Graves, A.-R. Mohamed and G. Hinton, *Speech Recognition with Deep Recurrent Neural*

Networks.

[23] A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, no. 4, pp. 664-676, April 2017.

[24] "Research Gate," [Online]. Available: https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=2ahUKEwimtefPidbaAhUF-2MKHX4IDukQjRx6BAgAEAU&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FRecurrent-versus-feedforward-neural-network_fig5_266204519&psig=AOvVaw3pAQrMlTBhPFAO_nlv8U.

[25] P. Wang, Y. Qian, F. K. Soong, L. He and H. Zhao, *Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network.*

[26] [Online]. Available: https://ytd2525.wordpress.com/2016/08/03/understanding-deriving-and-extending-the-lstm/.

[27] I. Sutskever, O. Vinyals and Q. V. Le, *Sequence to Sequence Learning with Neural Networks.*

[28] M. Ma, "A more detailed explaination about "the tensorflow chatbot"," 12 Oct 2016. [Online]. Available: https://github.com/Marsan-Ma/tf_chatbot_seq2seq_antilm/blob/master/README2.md.

[29] D. Britz, "Deep Learning for Chatbots, Part 1 – Introduction," April 2016. [Online]. Available: http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/.

[30] R. Yan, Y. Song and H. Wu, "Learning to Respond with Deep Neural Networks for Retrieval-Based Human-Computer Conversation System," in *Proceedings of the 39th International ACM SIGIR conference on research and development in information retrieval*, 2016.

[31] P. Surmenok, "Medium," September 2016. [Online]. Available: https://medium.com/@surmenok/chatbot-architecture-496f5bf820ed.

[32] "Cornell Movie-Dialogs Corpus," [Online]. Available: https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html.

[33] L. Khazan, "A Year of AI," 24 Apr 2016. [Online]. Available: https://ayearofai.com/lenny-2-autoencoders-and-word-embeddings-oh-my-576403b0113a.

[34] R. S. a. C. D. M. Jeffrey Pennington, "GloVe: Global Vectors for Word Representation," 2014.

[35] D. Britz, "WildML," 3 Jan 2016. [Online]. Available: http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/.

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

[37] M. L. Morales-Rodríguez, "ResearchGate," [Online]. Available: https://www.researchgate.net/figure/General-Diagram-of-the-AIML-Language-Extension_fig5_220887117.

[38] Kulbear, "Deep Learning Coursera," 13 Feb 2018. [Online]. Available: https://github.com/Kulbear/deep-learning-coursera/tree/master/Sequence%20Models/images.

[39] G. An, M. Shafiee and D. Shamsi, "Improving Retrieval Modeling Using Cross Convolution Networks And Multi Frequency Word Embedding," 2018. [Online]. Available: https://arxiv.org/pdf/1802.05373.pdf.

[40] S. Charrington, "What’s hot in AI: Deep reinforcement learning," VentureBeat, 5 Apr 2018. [Online]. Available: https://venturebeat.com/2018/04/05/whats-hot-in-ai-deep-reinforcement-learning/.