

Sequence Generation using Recurrent Neural Networks – A chat bot

DSE5499 – PROJECT WORK

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

Master of Technology

in

Data Science and Engineering

by

HARISH SUNDAR (15MDS0017)

Guided

by

PROF. KUMARAN. U



School of Information Technology and Engineering

February, 2017

Table of Contents:

Section ID	Section Name	Start Page	End Page
1	Introduction	2	4
2	Minimum System Requirements	5	5
3	Exiting System	6	6
4	Proposed System	7	7
5	Literature Survey	8	8
6	Core Concepts	9	16
7	System Architecture	17	20
8	Implementation	21	27
9	Future Work	28	28
10	References	29	30

Abstract

Our project's aim is to build a deep recurrent neural network (RNN) ^[R1] that reads an input file which consists of huge number of questions and answers. It uses the input file to get trained on all sort of conversations and generates an output that looks pretty much meaningful to the question we asked. Our main idea is to focus on training the recurrent neural network to understand the grammar, rhyming schemes ^[L1] like - assonance rhyme where only vowel sounds are shared, AAAA rhyme where every line rhymes and ABAB rhymes where every alternate line rhymes. The idea behind RNNs is to break the sequential information given as input and understand the semantics of input. A pure functional working procedure is that the trained RNNs predict an output word based on its previous word computations. We assume that RNNs have memory to remember all the computations it has done so far. Going deep we use LSTM (Long Short Term Memory) networks to generate sequences because they are very good in capturing the long-term semantics among all RNNs. To implement this type of network we use googles open sources deep learning library called "**Tensor Flow**" ^[L2].

1. INTRODUCTION:

Sequence generation using a chat bot serves as a platform where the customer can communicate with the chat bot which tries to reply as realistic as possible. The chat bot is trained on the Cornell movie dataset which consists of movie conversations, movie lines, movie characters and movie titles. The chat bot once trained tries to give a realistic response based on the conversations it has been trained when an input text is provided. Initially we used Hidden Markov model to train the chat bot on the movie conversations.

A brief description of the dataset used is given below:

1.1 Data

The input data is the form of several files such as movie characters, movie lines, movie conversations, movie titles. The data extraction is done by web data scraping using Python. A brief description of the data used is as follows:

- 220,579 conversational exchanges between 10,292 pairs of movie characters
- involves 9,035 characters from 617 movies
- in total 304,713 utterances
- movie metadata included:
 - genres
 - release year
 - IMDB rating
 - number of IMDB votes
 - IMDB rating
- character metadata included:
 - gender (for 3,774 characters)
 - position on movie credits (3,321 characters)

1.2 Scope:

The chat bot that we build will only be focusing on the textual conversations. The scope of our project will be restricted to textual conversations between the customer and organisation personnel in charge and we will not be focusing on the voice over communication between the customer and organisation personnel. We will extend our project to include all the voice-over communications made by the customers by converting them to textual conversations as earlier and then training our bot on the data.

1.3 Objective:

The main objective of our project would be to build an amazingly efficient chat bot which understands the input given by the user and tries to reply as realistic as possible so that the user understands what the chat bot is trying to convey. Ultimately, we will deploy the chat bot for the automation of helpline systems and support centre of the organisation. Our chat bot acts as a person in charge to help the customers in resolving their issues.

1.4 Motivation:

The idea of the project is originated from the following references - A Question answering network where the network is trained on the support Q&A data to generate auto responses based on the query, Image captioning network where a RNN is trained on various images with captions and asked to generate a caption for a new image, Machine generated city names where a RNN is trained on all USA City names to generate new city names. Also, we derived our idea from the – A chat bot replicating the artist's taste of lyrics where the chat bot is trained to generate lyrics of its own based on whatever lyrics it has been trained on. If the chat bot is trained using lyrics of a single artist, then it will try to preserve the rhyming scheme, taste of the artist while generating the output in a meaningful and concise manner. We are planning to build a chat bot which can be trained on a large set of conversations in the movie dataset and then can be used to answer all the questions we ask with the help of what it had learned during training. Then the chat bot can be used to chat with as a normal human being. We are

trying to model a chat bot which usually acts as a person in helping the customers to get their query resolved.

1.5 Problem Statement:

This projects aims to reduce the human effort of understanding the problem and come up with a solution. This system provides a better system translation and understanding of a user input. By having a better understanding, the system can provide even better response to the user that is close to human accuracy. To define the problem statement in detail, we use the following notions:

- T = training data
- E = Encoding data from the training data
- D = Decoding data from the training data
- F = Flow of encoding and the corresponding decoding data
- M = Hidden Markov Model built using the data ' T '
- U_{input} = User input for the system.
- n_{word} = Next word generated by using ' F ' with input as U_{input}
- $S_{response}$ = System response (Bag of meaningful n_{word} sets

By using the above notions, we can define the problem statement as the identification of appropriate decoding for the user input sentence and then continuously finding the next words until the response sentence is completed. i.e. Any question from the user will have an appropriate answer from the system based on the training data T . This system can be tested by any user based on the response that they get from the machine.

2. Minimum System Requirements

2.1 Hardware Requirements

Operating System	Linux (Preferable)
Primary Memory	4 GB (min.)
Secondary Memory	100 MB (min.)

2.2 Software Requirements

Name	Version
Python	2.6.6
D3 JS	4.2.2

Software	Packages
Python	Numpy
Python	Pandas
Python	Scikit-learn
Python	Scipy
Python	Tensor flow

3. Existing system:

The existing system of our project is a time-consuming process. The customers who are facing issues will be contacting the respective organisation where there will be a series of conversations doing rounds between the customer and organisation personnel in charge. The process of solving the issue works as follows: A customer facing an issue approaches the organisation's support centre through an email(say) which gets assigned to an organisation personnel who is in the support department. Then the personnel will start giving his solution through mail. The customer responds back with what has happened by his suggestion and the organisation personnel gets to know the likely cause of the issue and starts guiding him/her in the right direction towards the solution. Finally, the issue will be resolved with a series of conversations happened between the customer and organisation personnel. In this way of addressing queries, the customer will not be able to get rid of the issue in a short span of time. Immediate requirements of the customers cannot be handled by this approach efficiently even though the requirement is simple. This may cause a negative impact on the organisation to the customers. Another way of solving the issue immediately is by directly contacting the support centre via a phone call. In this way of approach, the customer could get his issue resolved in a shorter span of time than the other approach. Yet a series of conversations must be there via phone for the organisation personnel to get hold of the issue and guide the customer in the right direction. Most of the issues faced by the customers tends to be repetitive in time which gets assigned to the support personnel and takes more time to resolve the issue. These approaches incur the organisation a huge cost for solving the same issues again and again. We can collect all the conversations happened between the customer and organisation personnel in solving the issue either through call or mail and we might use this data to understand the common issues and solving them in no time for the customers. By this way, we are saving huge sum of revenue to the organisation in not investing much on the support department for solving the issues of the various customers.

4. Proposed System:

Our proposed system will be able to tackle the shortcomings of the existing system in an efficient manner. There will be a lot of conversations happening between customers and organisation personnel which we will use to understand the most frequently occurring issues and we will train a bot on these conversations to respond to the most frequently occurring issues with minimal or no human intervention. A chat bot, short for chat robot, is a computer program that tries to simulate human conversation or chat through artificial intelligence. Typically a chat bot will communicate with a real person and tries to communicate as realistic as possible. To reduce the cost for the organisation in the support department, we could collect the queries of different customers and their corresponding solutions to train our chat bot. The chat bot should be trained on huge number of conversations to be the most effective. When a new issue arises, which is different from the most common issues, the chat bot should be retrained on this new issue to effectively guide the customers even though they face a different issue. Our chat bot works as follows: The customer chats with our bot indicating his/her issue and the chat bot immediately provides the possible solution in no time and this conversation goes on until the issue of the customer gets resolved. Once our chat bot gets trained on the various conversations between customers and organisation personnel on the roads to solve the query, it can effectively suggest the new customers facing similar issues with less or no human intervention. Chat bots can be efficiently used for the automation of the customer support activities of various organizations. We can greatly reduce the cost of organisation in investing in the support department by deploying a chat bot in charge with minimal number of human resources. A single chat bot can handle hundreds of customers in guiding to their solutions helping them to effectively solve their issues in no time with less human intervention saving time and cost.

5. Literature survey:

Sequence generation using deep learning is an established concept that aims for better understanding of human to machine translation. In various industries, there are critical evaluations done on this topic but in a different field all together. Some of them are:

- 1) *Properties of Neural Machine Translation: Encoder-Decoder Approaches*, Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio: This paper aims to address the properties of machine translation using the deep learning technique called Recurrent Neural Networks (RNN). Our system uses this concept of machine translation by applying the encoder and decoder to encode the input sentence from the user and return an appropriate response using the decoder.
- 2) *Teaching Machines to Read and Comprehend*, Karl M. Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom, Google: This paper also aims to do end to end machine translation that can be helpful in the cases of speech-to-text conversion or language translations. We extend this system by applying these concepts to generate a bot that resolves the problems of user by taking their input into consideration.
- 3) *Dialog-based Language Learning*, Jason Weston, Facebook: In the paper, their system aims to achieve a bot that generates dialogues that are close to the trained movies. This makes the system learn the genre of the movie based on the dialogues and returns the best fit dialogues. We use the similar concept but in a different field with an aim to solve the user problems.

In all the above literatures, researchers define and explain the problem of machine translation using data in a structured format. They also use the concept of deep learning to improve the understanding of humans. To summarize the previous investigations to inform the readers, the current problem is integration of all the literatures to come up with a bot that resolves problems based on its knowledge.

6. Core concepts:

6.1 Machine Learning:

Machine learning is the art of getting computers to respond without being explicitly programmed. In the last decade, machine learning has provided us several stuffs like self-driving cars, practical speech recognition, effective web search and a vastly improved understanding of the human perception. Machine learning is so pervasive today that you probably use it several times a day without actually that you are using some sort of machine learning applications. Machine learning is a type of Artificial intelligence that provides computers with the capability to learn without any explicit programming to make it learn. The major focus of machine learning is the development of programs that can teach themselves to expand and change on seeing the new data. The process of machine learning is more similar to that of Data mining. Both the approaches search through the data to look for some well-defined patterns. However, instead of extracting data for human comprehension-as in the case of data mining applications-machine learning uses data to detect patterns in data and adjust program actions accordingly Machine learning algorithms are primarily classified into two types: Supervised machine learning and unsupervised machine learning. Supervised machine learning algorithms can use what it has learned in the past to predict the future. Unsupervised machine learning algorithms can draw inferences from the datasets without having any knowledge about the data.

6.2 Supervised Learning:

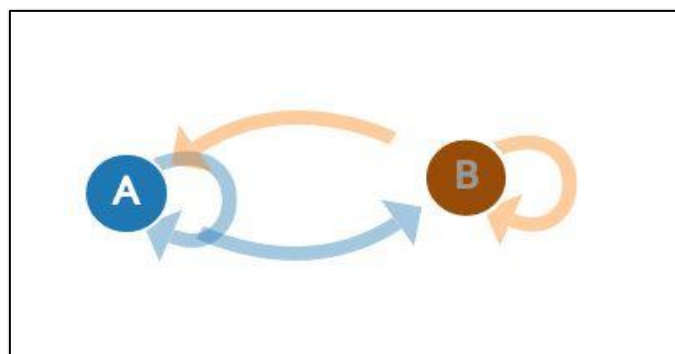
Supervised learning is a process of inferring a function from labelled training dataset. The training dataset contains a set of examples in which each example is labelled already. In supervised learning, each example is a pair consisting of an input vector and its desired output value. A supervised machine learning algorithm analyses the training data and produces an inferred function. This new inferred function can be used to map new examples. Generalization of the algorithm allows the algorithm to correctly label the new examples which it has never seen before.

6.3 Unsupervised Learning:





Unsupervised learning algorithms are used against data that has no historical labels associated with each of the training example. The system is not given the "right answer" or "right label" for each of the training example. The algorithm must figure out what is being shown. The goal is to explore the data and find some structure within the dataset provided without any prior information. Unsupervised learning works well on transactional data to find the structure within the data which cannot be identified otherwise. For example, it can identify segments of customers with similar attributes who can then be treated similarly in marketing campaigns of the organization. Or it can find the main attributes that separate customer segments from each other. Popular techniques include self-organizing maps, nearest-neighbour mapping, k-means clustering and singular value decomposition. These algorithms are also used to segment text topics, recommend items and identify data outliers. Clustering can be used to treat outliers in the data as well.

6.4 Markov Chains:

Markov chains, named after Andrey Markov, are mathematical systems that hop or transits from one "state" (a situation or set of values) to another. For example, if you made a Markov chain model of a child's behaviour , you might include "playing," "eating", "sleeping," and "crying" as states, which together with other behaviours could form a 'state space': a list of all possible states. In addition, on top of the state space, a Markov chain tells you the probability of "hopping", or "transitioning," from one state to any other state---e.g., the chance that a baby currently playing will fall asleep in the next five minutes without crying first.



With two states (A and B) in our state space, there are 4 possible transitions or hopping (not 2, because a state can transition back into itself). If we're at 'A' we could transition to 'B' or stay at 'A'. If we're at 'B' we could transition to 'A' or stay at 'B'. In this two state diagram, the probability of transitioning from any state to any other state is 0.5 because the state can either transit to another state or into itself. Sometimes, real modelers don't always draw out Markov chain diagrams. Instead they use a "transition matrix" to tally the transition probabilities. Every state in the state space is included once as a row and again as a column, and each cell in the matrix tells you the probability of transitioning from its row's state to its column's state. So, in the matrix, the cells do the same job that the arrows do in the diagram.

	A	B
A	$P(A A):$ 0.50 	$P(B A):$ 0.50 
B	$P(A B):$ 0.50 	$P(B B):$ 0.50 

If the state space adds one state, we add one row and one column, adding one cell to every existing column and row. This means the number of cells grows quadratically as we add states to our Markov chain. Thus, a transition matrix comes in handy pretty quickly, unless you want to draw a jungle gym Markov chain diagram.

One use of Markov chains is to include real-world phenomena in computer simulations. For example, we might want to check how frequently a new dam will overflow, which depends on the number of rainy days in a row. To build this model, we start out with the following pattern of rainy (R) and sunny (S) days:

R R R R R S S S S S S S S R R R R R.....

One way to simulate this weather would be to just say "Half of the days are rainy. Therefore, every day in our simulation will have a fifty percent chance of rain." This rule would generate the following sequence in simulation:

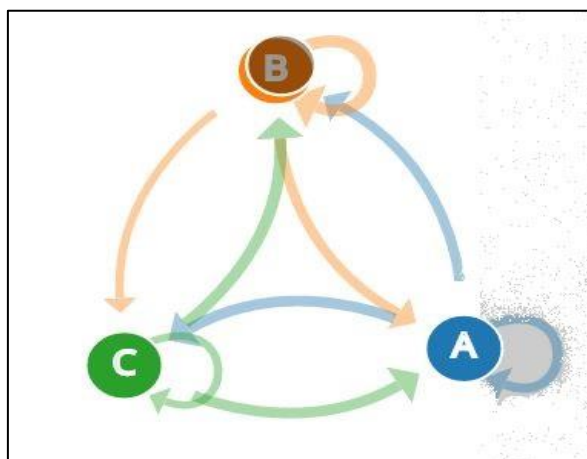
R R R R S R S R R S S S R S R S R S

The second sequence seems to jump around, while the first one (the real data) seems to have a "stickiness". In the real data, if it's sunny (S) one day, then the next day is also much more likely to be sunny.

We can mimic this "stickiness" with a two-state Markov chain. When the Markov chain is in state "R", it has a 0.9 probability of staying put and a 0.1 chance of leaving for the "S" state. Likewise, "S" state has 0.9 probability of staying put and a 0.1 chance of transitioning to the "R" state.

In the hands of meteorologists, ecologists, computer scientists, financial engineers and other people who need to model big phenomena, Markov chains can get to be quite large and powerful. For example, the algorithm Google uses to determine the order of search results, called "PageRank", is a type of Markov chain.

A more complex Markov chain:



6.5 Hidden Markov model:

A Hidden Markov model is a tool for representing probability distributions over sequences of observations. Let us denote the observation at time 't' by the variable Y_t . This can be a symbol from a discrete alphabet, a real valued variable, an integer, or any other object, if we can define probability distributions over it. Hidden Markov models are almost used in all current speech recognition systems, in numerous applications in computational molecular biology, in data compression and in other areas of Artificial intelligence and pattern recognition. Hidden Markov models have found their place in the field of Computer vision applications. *Markov processes* are examples of stochastic processes—processes that generate random sequences of outcomes or states according to certain probabilities. Markov processes are distinguished by being memoryless—their next state depends only on their current state, not on the history that led them there. Models of Markov processes are used in a wide variety of applications, from daily stock prices to the positions of genes in a chromosome. *Hidden Markov Models (HMM)* seek to recover the sequence of states that generated a given set of observed data.

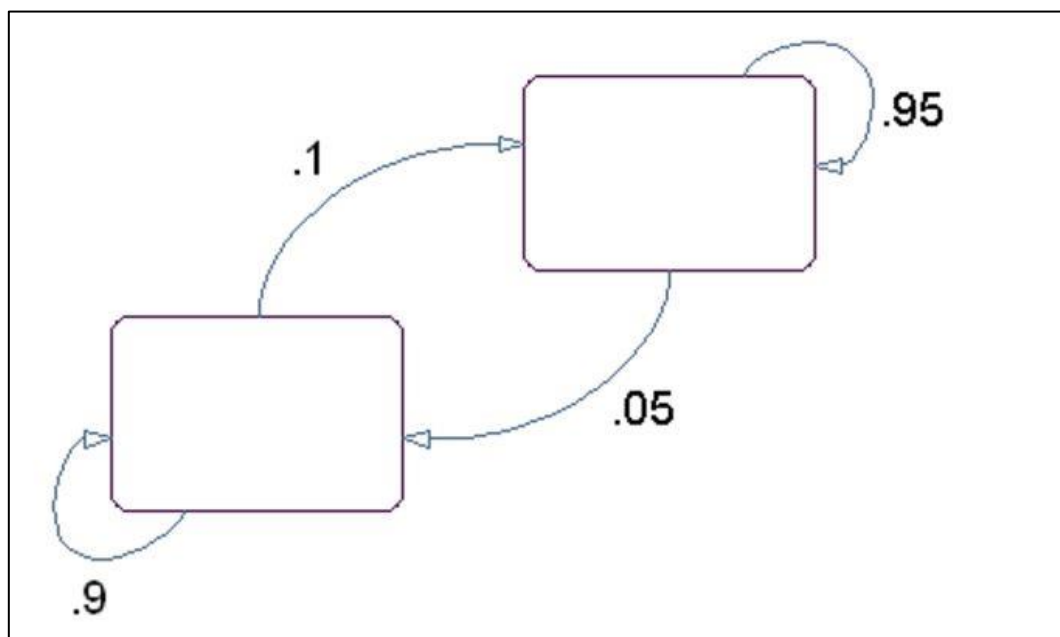
A *hidden Markov model* (HMM) is one in which you observe a sequence of emissions, but do not know the sequence of states the model went through to generate the emissions. Analyses of hidden Markov models seek to recover the sequence of states from the observed data.

As an example, consider a Markov model with two states and six possible emissions. The model uses:

- A red die, having six sides, labeled 1 through 6.
- A green die, having twelve sides, five of which are labeled 2 through 6, while the remaining seven sides are labeled 1.
- A weighted red coin, for which the probability of heads is .9 and the probability of tails is .1.
- A weighted green coin, for which the probability of heads is .95 and the probability of tails is .05.

The model creates a sequence of numbers from the set $\{1, 2, 3, 4, 5, 6\}$ with the following rules:

- Begin by rolling the red die and writing down the number that comes up, which is the emission.
- Toss the red coin and do one of the following:
 - If the result is heads, roll the red die and write down the result.
 - If the result is tails, roll the green die and write down the result.
- At each subsequent step, you flip the coin that has the same color as the die you rolled in the previous step. If the coin comes up heads, roll the same die as in the previous step. If the coin comes up tails, switch to the other die.
- The state diagram for this model has two states, red and green, as shown in the following figure.



You determine the emission from a state by rolling the die with the same color as the state. You determine the transition to the next state by flipping the coin with the same color as the state.

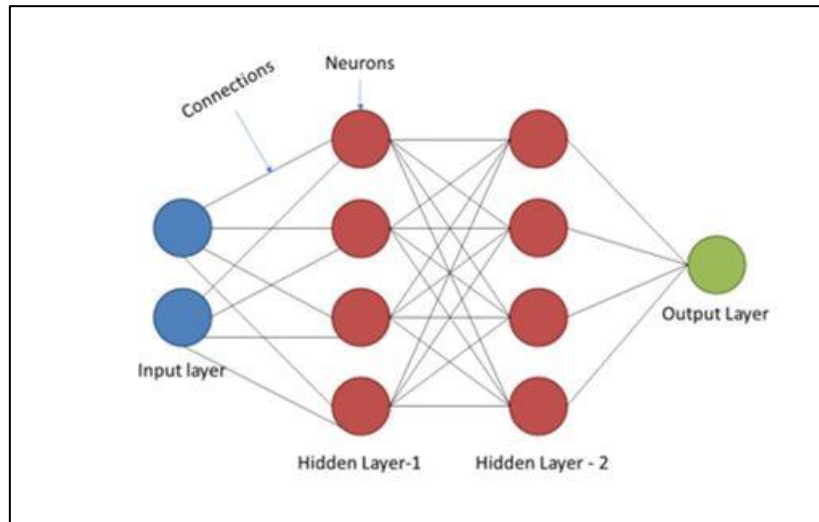
The model is not hidden because you know the sequence of states from the colors of the coins and dice. Suppose, however, that someone else is generating the emissions without showing you the dice or the coins. All you see is the sequence of emissions. If you start seeing more 1s than other numbers, you might suspect that the model is in the green state, but you cannot be sure because you cannot see the color of the die being rolled. Hidden Markov models are used to generate text based on the large number of texts it has been trained with. In our case, we will train the Hidden markov model with our movie dataset to generate some sentences as realistic as possible. Ultimately, we will train a Recurrent neural network with our movie dataset which has a large list of conversations. Then the network will be able to reply as realistic as possible to the text input which has been provided.

6.6 Neural Networks (NN):

NN is defined by **Dr. Robert Hecht-Nielsen** (inventor of first neuro computers) as "*A computing system made up of several simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.*" In general, NNs are referred to human brain as their functional behaviour resembles to our brain. Typical neural networks are organized in the form of three layers

- a) Input layer
- b) Hidden layers
- c) Output layers

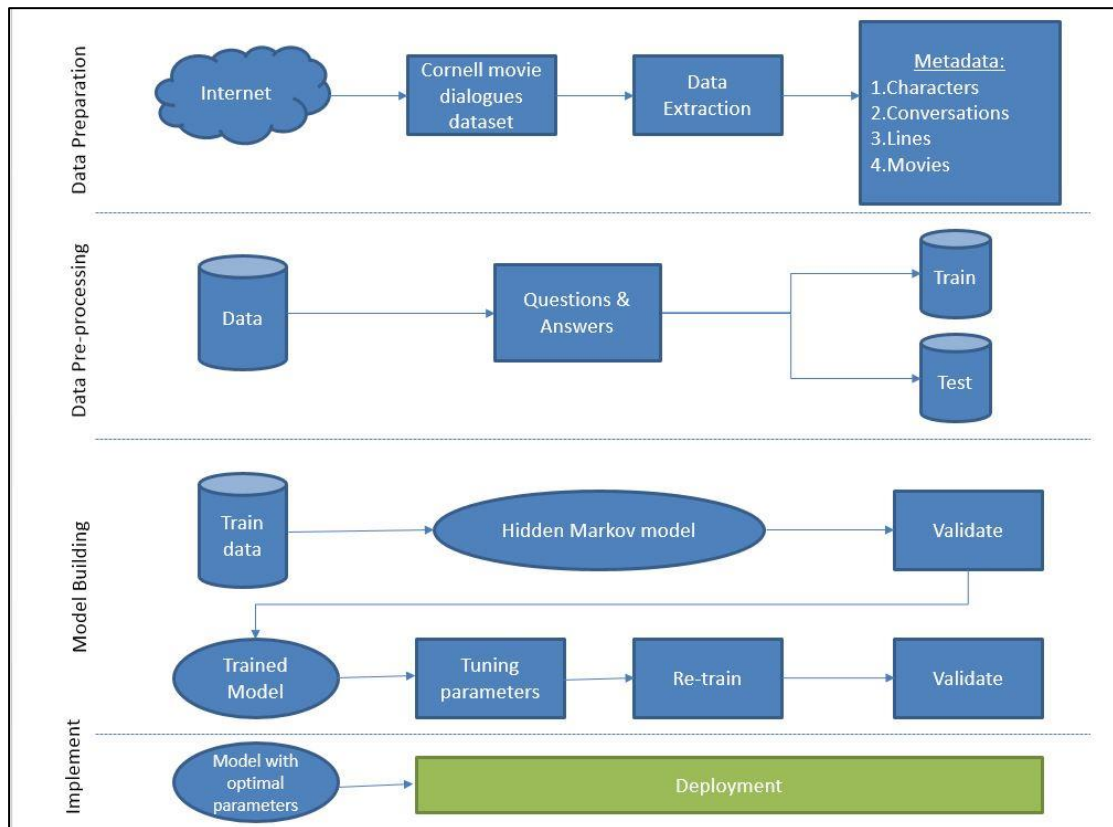
and each hidden layer will have multiple number of interconnected nodes which contain a function to process the signal from input and this function is called activation function. The process of classification in ANN is achieved in the following way. The input node transmits the data point through the hidden layers where multiple processing steps are done on the point and a final output is achieved at the output layer. Then, the output achieved is compared with the original output and the error is back propagated through the network which causes the adjustment of weights in the hidden layers. i.e. most ANNs formulate a simple learning rule that modifies weights of the connections based on the error propagated backwards. This process is performed multiple times until a minimized error is achieved. The functioning of a neural network is represented in the below diagram.



An important application of neural networks is pattern recognition. Pattern recognition can be implemented by using a feed-forward (figure 1) neural network that has been trained accordingly. During training, the network is trained to associate outputs with input patterns. When the network is used, it identifies the input pattern and tries to output the associated output pattern. The power of neural networks comes to life when a pattern that has no output associated with it, is given as an input. In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern. Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.

7. SYSTEM ARCHITECTURE:

The architecture which we have followed in our project is as follows:



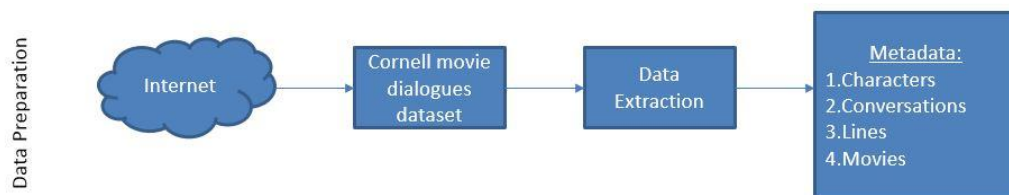
In our architecture, we have basically four layers which we will follow up to complete the project. The four layers are

1. 1.Data Scraping
2. 2.Data Pre-processing
3. 3.Model building and Tuning
4. 4.Implementation

In data scraping layer, we will be scraping our required dataset in the web using some python scripts. In the data pre-processing layer, we will be using the dataset which we collected in the data scraping layer to process it to be fed in to the model of interest. In the model building and Tuning layer, we will be building the model with the data we have and try to tune up parameters. At last, we will implement our model to production. A detailed review of each of these layers are given below:

7.1 Data Preparation:

In this layer the system fetches the data of movie conversations from the website of Cornell university. It contains a large amount of meta-data and fictional conversations extracted from raw movie scripts. The downloaded file will be a compressed format with an extension of (.zip). We need to extract all the files inside the compressed folder into a physical folder before applying the pre-processing step. The extracted data consists of meta-data and details about the characters in the movie, conversations happened in the movie, line numbers that map back to the conversations, and movie titles and other meta-data. This data is processed to obtain a flow of encoder and decoder which can then be used for building a deep neural network called Recurrent Neural Network. The data preparation layer can be represented as a flow chart as shown below.



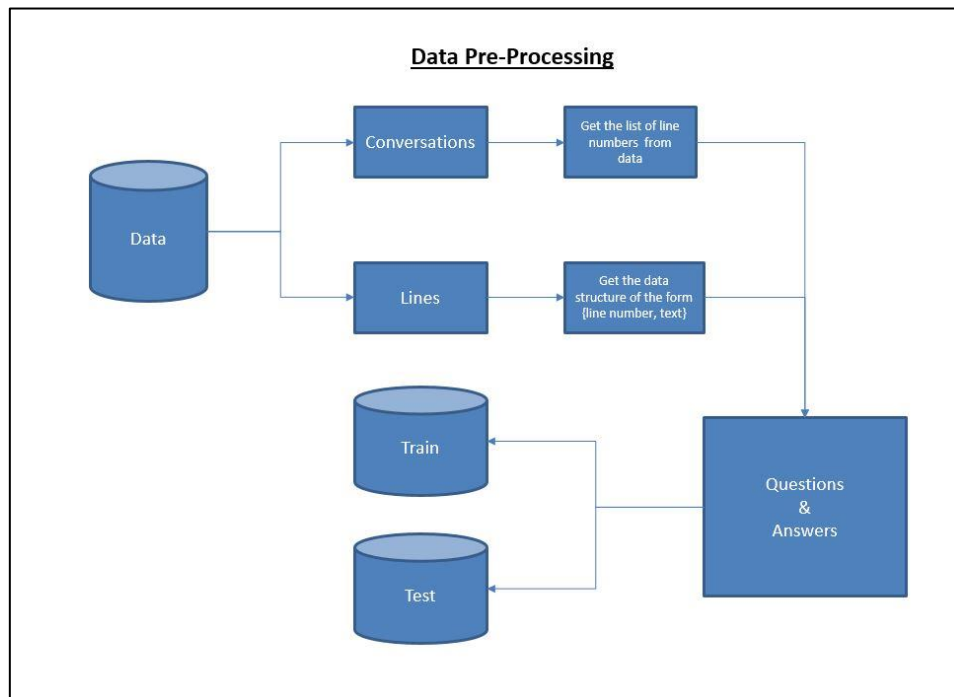
7.2 Data Pre-processing:

The movie dataset has several files including movie conversations, movie lines, movie characters, movie titles. We need to process these several files into a single file which we will feed to our model which will be ultimately Recurrent Neural networks.

The data which we are using should be prepared in a specific format. Movie lines file contains line id, movie id, character id, character name and the corresponding text. Using this file, we need to extract the line id and its corresponding text in the form of key – value pairs. We accomplished this using a simple python script. The movie conversation file contains all the line ids which makes up each conversation.

We need to take all the line ids from the conversations file and append it to a single list. We will use a python script to create a list of conversations each containing the line ids

of the corresponding conversation. Using the two intermediate files we created, we can use them to create a sequence or list of questions and answers which we will ultimately feed into our model of interest which may be Hidden Markov model or Recurrent neural networks (RNN).



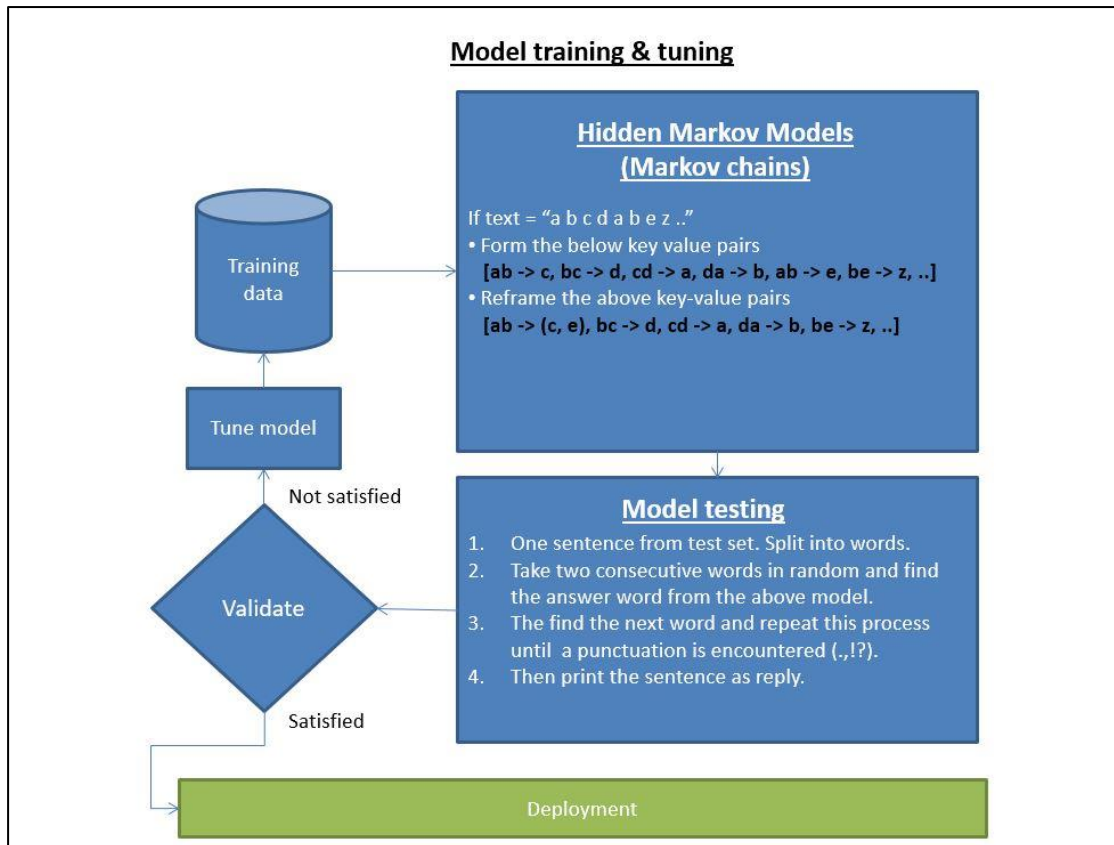
7.3 Model building and Tuning:

The model building process involves setting up ways of collecting data, understanding and paying attention to what is important in the data to answer the questions you are asking, finding a statistical, mathematical or a simulation model to gain understanding and make predictions.

All of these things are equally important and model building is a crucial skill to acquire in every field of science. The process stays true to the scientific method, making what you learn through your models useful for gaining an understanding of whatever you are investigating as well as make predictions that hold true to test. This process involves asking questions, gathering and manipulating data, building models, and ultimately testing and evaluating them. In our project, we are building a Hidden Markov model on

the movie conversations data. Once the model has been built, we will check the performance of the model by giving some text as input and seeing how realistic outputs we are getting based on the input provided.

If we are not satisfied with the performance of the model, we could play around by tuning various parameters of the model and rebuilding the model from scratch until we get a model which is giving as realistic replies as possible.



7.4 Implementation Layer:

Once the model has been trained and it is able to produce as realistic replies as possible, then we can deploy our model to production for various applications. In our project, we can deploy our chat bot to reply in the form of conversations on which it has been trained for days. In future, we are planning to present an interactive user interface where we can be able to communicate with the chat bot in a smooth manner

8. IMPLEMENTATION:

Implementation of the chat bot goes layer by layer as specified in the system architecture. Each layer will be implemented in a sequential manner.

8.1 Data Preparation:

The data which we will use is the Cornell movie dataset which we will get using a python script. The movie dataset basically has four files which we could make use of to build our model on the data. They are as follows:

1. Movie conversations
2. Movie lines
3. Movie characters
4. Movie titles

And several other metadata which we are least bothered of. A glimpse of each of the dataset will be able to provide a clarity on our understanding of the data. Movie conversations dataset contains basically the line numbers constituting each of the conversations with movie id, user id separated by the special character “+++\$\$+++”

```
u0 +++$$+++ u2 +++$$+++ m0 +++$$+++ ['L367', 'L368']
u0 +++$$+++ u2 +++$$+++ m0 +++$$+++ ['L401', 'L402', 'L403']
u0 +++$$+++ u2 +++$$+++ m0 +++$$+++ ['L404', 'L405', 'L406', 'L407']
u0 +++$$+++ u2 +++$$+++ m0 +++$$+++ ['L575', 'L576']
u0 +++$$+++ u2 +++$$+++ m0 +++$$+++ ['L577', 'L578']
u0 +++$$+++ u2 +++$$+++ m0 +++$$+++ ['L662', 'L663']
```

Movie lines dataset contains basically the line IDs and their corresponding text pertaining to the line ID with other information such as user ID, movie ID, user name separated by the special character “+++\$\$+++”.


```

L1045 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ They do not!
L1044 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ They do to!
L985 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ I hope so.
L984 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ She okay?
L925 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ Let's go.
L924 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ Wow
L872 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ Okay -- you're gonna need to learn how to lie.
L871 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ No

```

Movie characters dataset contains the character names and the movie names with their corresponding IDs

```

u0 +++$+++ BIANCA +++$+++ m0 +++$+++ 10 things i hate about you +++$+++ f +++$+++ 4
u1 +++$+++ BRUCE +++$+++ m0 +++$+++ 10 things i hate about you +++$+++ ? +++$+++ ?
u2 +++$+++ CAMERON +++$+++ m0 +++$+++ 10 things i hate about you +++$+++ m +++$+++ 3
u3 +++$+++ CHASTITY +++$+++ m0 +++$+++ 10 things i hate about you +++$+++ ? +++$+++ ?
u4 +++$+++ JOEY +++$+++ m0 +++$+++ 10 things i hate about you +++$+++ m +++$+++ 6

```

Movie titles dataset contains basically the title of the movie with its ID associated separated by the special character “+++\$+++”

```

m9 +++$+++ the atomic submarine +++$+++ 1959 +++$+++ 4.90 +++$+++ 513 +++$+++ ['sci-fi', 'thriller']
m10 +++$+++ affliction +++$+++ 1997 +++$+++ 6.90 +++$+++ 7252 +++$+++ ['drama', 'mystery', 'thriller']
m11 +++$+++ air force one +++$+++ 1997 +++$+++ 6.30 +++$+++ 61978 +++$+++ ['action', 'drama', 'thriller']
m12 +++$+++ airplane ii: the sequel +++$+++ 1982 +++$+++ 5.80 +++$+++ 15210 +++$+++ ['comedy', 'romance', 'sci-fi']
m13 +++$+++ airplane! +++$+++ 1980 +++$+++ 7.80 +++$+++ 57692 +++$+++ ['comedy', 'romance']
m14 +++$+++ alien nation +++$+++ 1988 +++$+++ 6.10 +++$+++ 5590 +++$+++ ['crime', 'drama', 'sci-fi', 'thriller']
m15 +++$+++ aliens +++$+++ 1986 +++$+++ 8.50 +++$+++ 173518 +++$+++ ['action', 'sci-fi', 'thriller']

```

Next, we would be using the gathered data to frame list of questions and answers to train our model. We used Python to generate the dataset in the desired format.

8.2 Data Pre-processing:

The movie dataset has several files including movie conversations, movie lines, movie characters, movie titles. We need to process these several files into a single file which we will feed to our model which will be ultimately Recurrent Neural networks.

In this layer, we will be performing several merge operations between files to get our data in the desired format. We will first get a list of line id and its corresponding text in the form of key-value pairs

```

'''
    1. Read from 'movie-lines.txt'
    2. Create a dictionary with ( key = line_id, value = text )
'''
def get_id2line():
    lines=open('movie_lines.txt').read().split('\n')
    id2line = {}
    for line in lines:
        _line = line.split(' +++$+++ ')
        if len(_line) == 5:
            id2line[_line[0]] = _line[4]
    return id2line

```

Then we will be processing the data to get a list of line ids of each conversation. The code snippet for achieving the same would be

```

'''
    1. Read from 'movie_conversations.txt'
    2. Create a list of [list of line_id's]
'''
def get_conversations():
    conv_lines = open('movie_conversations.txt').read().split('\n')
    convs = [ ]
    for line in conv_lines[:-1]:
        _line = line.split(' +++$+++ ')[-1][1:-1].replace('"', "").replace(" ", "")
        convs.append(_line.split(','))
    return convs

```

Using the two intermediate files we created, we can use them to create a sequence or list of questions and answers which we will ultimately feed into our model of interest which may be Hidden Markov model or Recurrent neural networks(RNN). The questions file will contain the starting text of each of the conversation and every alternate text in each conversation. The answers file will be having the rest of the text in it which is not there in the questions file. The code snippet to achieve the same would be

```

'''
    Get lists of all conversations as Questions and Answers
    1. [questions]
    2. [answers]
'''
def gather_dataset(convs, id2line):
    questions = []; answers = []

    for conv in convs:
        if len(conv) %2 != 0:
            conv = conv[:-1]
        for i in range(len(conv)):
            if i%2 == 0:
                questions.append(id2line[conv[i]])
            else:
                answers.append(id2line[conv[i]])

    return questions, answers

```

Finally, we will be dividing our processed file into training and testing data. The training data will be used for training the model and test data will be used to test the performance of the bot we have built with the training data.

```

def prepare_seq2seq_files(questions, answers, path='', TESTSET_SIZE = 30000):

    # open files
    train_enc = open(path + 'train.enc', 'w')
    train_dec = open(path + 'train.dec', 'w')
    test_enc = open(path + 'test.enc', 'w')
    test_dec = open(path + 'test.dec', 'w')

    # choose 30,000 (TESTSET_SIZE) items to put into testset
    test_ids = random.sample([i for i in range(len(questions))], TESTSET_SIZE)

    for i in range(len(questions)):
        if i in test_ids:
            test_enc.write(questions[i] + '\n')
            test_dec.write(answers[i] + '\n')
        else:
            train_enc.write(questions[i] + '\n')
            train_dec.write(answers[i] + '\n')
        if i%10000 == 0:
            print '\n>> written %d lines' %(i)

    # close files
    train_enc.close()
    train_dec.close()
    test_enc.close()
    test_dec.close()

```

8.3 Model Building and Tuning:

Using the data which we have processed, we can start building the model on the training data. We have accomplished building the model using Python. We are building a Hidden Markov model to get trained on the dataset which we created in the earlier steps. Hidden Markov models are giving a decent reply to each of the conversation we input but it is not up to the mark. Ultimately, we are planning to implement the recurrent neural network which is extremely useful for generating meaningful sequence of conversation. We have created two scripts in Python in which one script will be used to train the model and another script will be used to test the model. The code snippet for training the model is given below:

```

'''
import the modules needed
'''
import pickle

'''
read the file with questions [The encoding part]
'''
questions = open('questions.txt')
qData = questions.read()
# Split the conversation into lines
qLines = qData.splitlines()
qWords = [eachLine.split() for eachLine in qLines]
questions.close()

'''
read the file with answers [The decoding part]
'''
answers = open('answers.txt')
aData = answers.read()
# Split the conversation into lines
aLines = aData.splitlines()
aWords = [eachLine.split() for eachLine in aLines]
answers.close()

'''
qaFlow = dictionary with question word as key and immediate
answer word mapping as value
'''
qaFlow = {}
for l in range(len(qLines)):
    working = []
    check = qWords[l]
    for w in range(len(aWords) - 1):
        if check == aWords[w] and aWords[w][-1] not in '(),.?!':
            working.append(str(aWords[w + 1]))
    qaFlow[check] = working
    if l % 100 == 0:
        print ">> Trained on " + str(l) + " sentences".

'''
write the list object of Q&A mapping to a pickle
'''
f = open('qa-pickle','wb')
pickle.dump(qaFlow, f, 2)
f.close()

```

Similarly, we have created another script for testing the model by giving some textual input to the model and seeing their responses. The code snippet for testing the model is given below:

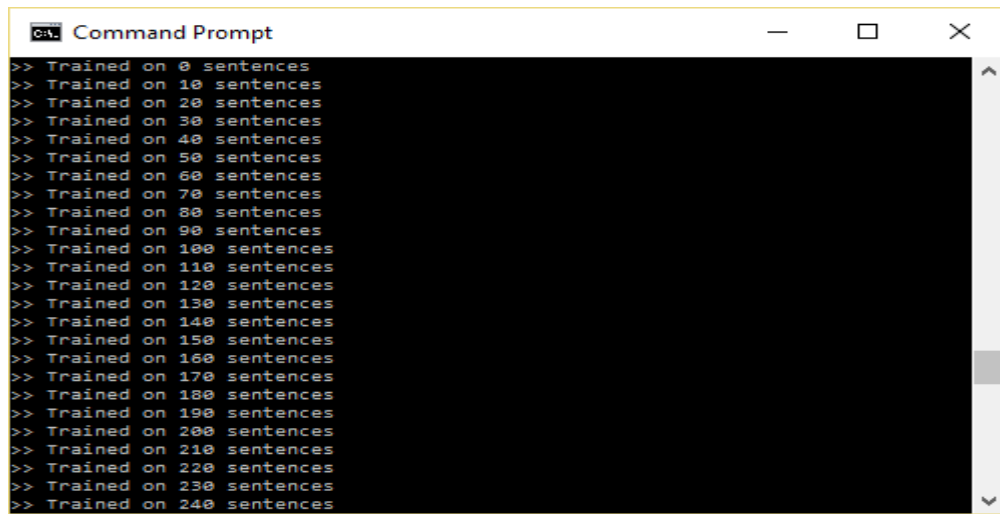
```
"""
import the modules needed
"""
import pickle
import random

"""
open the Q & A flow object and read it in the console
"""
f = open('qa-pickle', 'rb')
qaFlow = pickle.load(f)
f.close()

"""
define a function that selects the next word using
the Q & A flow.
If nothing is found it returns 'the'
"""
def nextword(a):
    if a in qaFlow:
        return random.choice(qaFlow[a])
    else:
        return 'the'

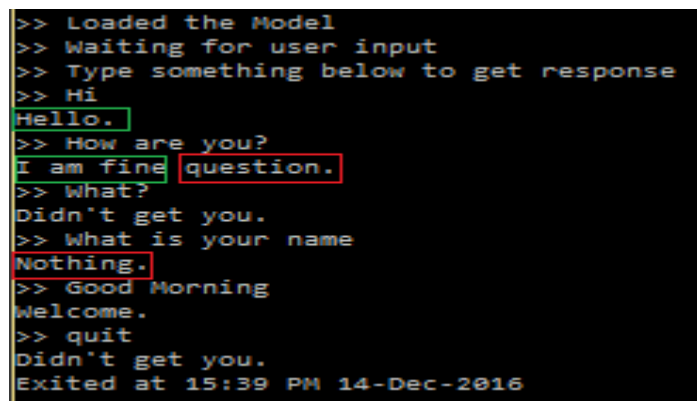
"""
In main function, take the user input and select a
random word from his input to select the next word.
this continues until there is a punctuation. {,?!}
"""
if __name__ == "__main__":
    speech = ''
    print ">> Loaded the Model"
    print ">> Waiting for user input"
    print ">> Type something below to get response"
    while speech != 'quit':
        speech = raw_input('>>')
        s = random.choice(speech.split())
        response = ''
        while True:
            newword = nextword(s)
            response += ' ' + newword
            s = newword
            if newword[-1] in ',?!.':
                break
        print response
```


The sample output during training phase is given below



```
C:\> Command Prompt
>> Trained on 0 sentences
>> Trained on 10 sentences
>> Trained on 20 sentences
>> Trained on 30 sentences
>> Trained on 40 sentences
>> Trained on 50 sentences
>> Trained on 60 sentences
>> Trained on 70 sentences
>> Trained on 80 sentences
>> Trained on 90 sentences
>> Trained on 100 sentences
>> Trained on 110 sentences
>> Trained on 120 sentences
>> Trained on 130 sentences
>> Trained on 140 sentences
>> Trained on 150 sentences
>> Trained on 160 sentences
>> Trained on 170 sentences
>> Trained on 180 sentences
>> Trained on 190 sentences
>> Trained on 200 sentences
>> Trained on 210 sentences
>> Trained on 220 sentences
>> Trained on 230 sentences
>> Trained on 240 sentences
```

Similarly, the sample output during testing the model is given as follows:



```
>> Loaded the Model
>> Waiting for user input
>> Type something below to get response
>> Hi
Hello.
>> How are you?
I am fine question.
>> What?
Didn't get you.
>> What is your name
Nothing.
>> Good Morning
Welcome.
>> quit
Didn't get you.
Exited at 15:39 PM 14-Dec-2016
```

9. Future work:

As discussed in the above sections, we have only implemented the base system for chat bot. There are many updates to be done to improve the bot performance. Some of them are:

- a) Dive Deep: Instead of trying a base Markov chain model, we will try the state-of-the-art Recurrent Neural Networks (RNN). This should result in a performance that can scale big and that can beat any other perfectly hand-crafted model. We especially use Long and Short Term Memory (LSTM) networks that maintains a memory of the conversation so it will not just be a question-answer bot, but it will be a conversational bot.
- b) Use FAQ posts: Instead of training the bot on the movie conversations, we can get the data of all the questions posted by different users in various forums. We can use this data to train and test. This will help to achieve our purpose of solving the user problems.
- c) Re-training: We can ask for the user feedback to a mis-classified question and then assign more weight to the solution before re-training the network. This additional weight causes more swing in the network towards positive direction making the network adjusting itself to this problem. Thus, the next time the network tries not to repeat the mistake.
- d) Use GPU: All the concepts of deep learning are usually better implemented over GPU than a CPU. This helps the algorithm scale better and scale faster. This will reduce the time of training and also reduces the stress on the CPU cores. Usually with a large dataset of movie dialogues, the training time on CPU will take 2 days where the usage of GPU takes less than 5 hours.
- e) Article writing: The same network trained can be used to write articles in a blog. The LSTM / HMM will need a training data of all the articles on a certain topic. The seq2seq model from Googles Tensor Flow module can be used to implement this Deep Writing.

10. References

Web links

1. <http://literarydevices.net/rhyme-scheme/>
2. <https://www.tensorflow.org/>
3. https://github.com/tflearn/tflearn/blob/master/examples/nlp/lstm_generator_cityname.py

Research Papers

1. *Hasim Sak, Andrew Senior, Francoise Beaufays - "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modelling".*
2. *Di Wang and Eric Nyberg - "A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering".*
3. *Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan, Google - "Show and Tell: A Neural Image Caption Generator".*
4. *Karteek Addanki, Dekai Wu - "Unsupervised Rhyme Scheme Identification in Hip Hop Lyrics Using Hidden Markov Models".*
5. *Chappie - A Semi-automatic Intelligent Chatbot by Bibek Behera in Learning, Consumer products, Screens, Text recognition, Lexical formulation, Sales (2016)*
6. *Hedonic VS Utilitarian Values of Online Avatars by Reza Etemad in Applications, Contact center integration (2015)*
7. *Smart Language Morphologic Analyzer-Tagger by Andres Hohendahl in Agent's perception of humans, Context, Text recognition, Relations, Learning, Cognition, Back End Integration and Knowledge bases, User Client Technology, Consumer products, E-Learning, Front End Integration, Products, Contact center integration (2015)*
8. *Commercial chatbot: performance evaluation, usability metrics, and quality standards of ECA by Karolina Kuligowska in (2015)*
9. *Narrating the Opac: How Storytelling and Narrative by Mark-Shane Scale and Anabel Quan-Haase in Agent's Capabilities, Role playing, Lookup (2013)*

10. *Chatbots in the library: is it time?* by DeeAnn Allison in *E-Learning, Coach* (2012)
11. *Towards a Game-Chatbot* by Peter van Rosmalen, Johan Eikelboom, Erik Bloemers, Kees van Winzum and Pieter Spronck in *Agent identity, Knowledge, Applications, Back End Integration and Knowledge bases, Emotion, Learning, Cognition* (2012)
12. *Conversation in HMM-based speech synthesis* by Sebastian Andersson, Junichi Yamagishi and Robert A.J. Clark in *Emotion, Speech synthesis (TTS)* (2012)
13. *Virtual Human Dialogue System Speech Understanding* by David DeVault and David Traum in *Context, Speech recognition, Text recognition* (2012)
14. *Evaluation of Virtual Human Dialogue Systems* by Kallirroi Georgila, Alan W. Black, Kenji Sagae and David Traum in *Speaker recognition, Speech recognition, Speech synthesis (TTS)* (2012)
15. *Human-Level Artificial General Intelligence* by Sam S. Adams, Itamar Arel, Joscha Bach, Robert Coop, Rod Furlan, Ben Goertzel, J. Storrs Hall, Alexei Samsonovich, Matthias Scheutz, Matthew Schlesinger, Stuart C. Shapiro and John Sowa in *Agent's Processing, Learning, Knowledge, Psychology* (2011)
16. *Teacher Agents* by Kevin Reed and Gabriele Meiselwitz in *Coach* (2011)
17. *Long-Term Human-Robot Interaction* by Nikolaos Mavridis, Michael Petychakis, Alexandros Tsamakos, Panos Toulis, Shervin Emami, Wajahat Kazmi, Chandan Datta, Chiraz BenAbdelkader and Andry Tanoto in *Face recognition* (2011)
18. *A platform for managing campaigns over Social Media* by Michael Petychakis, Iosif Alvertis, George Gionis and Robert Kleinfeld in *User Client Technology* (2011)
19. *Artificially intelligent conversational agents in libraries* by Victoria L. Rubin, Yimin Chen and Lynne Marie Thorimbert in *Agent's Capabilities, Coach* (2010)