CHATBOT

Pranitha Keerthi, Saivenkat Thatikonda, and Twisha Purohit

Abstract Chatbots are one class of intelligent, conversational software agents activated by natural language input (which can be in the form of text, voice, or both). They provide conversational output in response, and if commanded, can sometimes also execute tasks. In this project, we try to build a general chatbot that can have a general conversation with us like a friend instead of a specified chat for some specific purpose. To train this chatbot, Cornell Movie- Dialogs Corpus was used. This dataset consists of more than 600 movies containing thousands of conversations between tons of characters. The dataset is first cleaned to help in better training and then we build Sequence2Sequence model which is a combination of two recurrent neural network kept in sequence.

I. Introduction

A chatbot is artificial intelligence (AI) software that can make a conversation with a user in natural language through messaging applications, websites, and mobile apps or through the telephone. A chatbot is frequently depicted as a standout amongst the most progressive and promising articulations of connection among people and machines. The service could be any number of things, ranging from functional to fun, and it could live in any major chat product (Facebook Messenger, Slack, Telegram, Text Messages, etc.). Be that as it may, from an innovative perspective, a chatbot only represents the natural evolution of a Question Answering system leveraging Natural Language Processing (NLP). Detailing reactions to inquiries in regular language is a standout amongst the most commonplace Instances of Natural Language Processing connected in different ventures' end-use applications.

Chatbot applications streamline interactions among individuals and services, upgrading client experience. In the meantime, they offer organizations new chances to improve the clients' commitment process and operational effectiveness by lessening the normal expense of customer service. To be fruitful, a chatbot ought to have the capacity to adequately perform both of these errands. Human help assumes a key job here: Paying little heed to the sort of methodology and the stage, human intervention is critical in designing, preparing and improving the chatbot framework.

Problem Statement: The ability to identify the user's intent and extract data and relevant entities contained in the user's request is the first problem and the most relevant step at the core of a chatbot: If we are not able to correctly understand the user's request, we won't be able to provide the correct answer. Returning the response: once the user's intent has been identified, the chatbot must provide the most appropriate response for the user's request. The answer may be:

- a generic and predefined text
- a text retrieved from a knowledge base that contains different answer
- a contextualized piece of information based on data the user has provided data stored in enterprise systems
- the result of an action that the chatbot performed by interacting with one or more backend application
- a disambiguating question that helps the chatbot to correctly understand the user's request

Goal: To build a chatbot that can have basic, meaningful conversation with the end user using recurrent neural networks.

II. Motivation

There are a wide range of uses of chatbots in many different kinds of industries. Chatbots save time and efforts by automating customer support. The opportunities provided by chatbot systems go far beyond giving responses to customers' inquiries. They are also used for other business tasks, like collecting information about users, helping to organize meetings and reducing overhead costs.

Business need chatbots for reasons like getting rid of routine tasks and simultaneous processing of multiple requests from users. Besides, a tremendous speed of processing users' requests with chatbots helps gaining customers' loyalty. Some of the applications are:

- Available whenever
- Dealing with Limit
- Adaptable quality
- Consumer loyalty
- Savvy
- Quicker On boarding
- Work Robotization
- Personal Assistant

1

III. Related Work

A chatbot resembles an ordinary application. There is an application layer, a database and APIs to call other outer organizations. Clients can without much of a stretch access chatbots; it adds unpredictability for the application to deal with. Notwithstanding, there is a typical issue that must be handled. It can't understand the arrangement of the client. Right now, bots are prepared by the past data accessible to them. Thus, most associations have a chatbot that keeps up logs of talks. Engineers use these logs to dissect what customers are attempting to inquire. With a mix of AI devices and models, designers arrange customer request and answer with the best fitting answer. For instance, if any client is getting some information about instalments and receipts, for example, "where is my item instalment receipt?" and "I haven't gotten an instalment receipt?", the two sentences are taken to have a similar importance. On the off chance that there is no thorough information accessible, at that point diverse APIs can be used to prepare the chatbot.

The Chatbots work based on three classification methods: Pattern Matches, Natural Language Understanding (NLU) and Natural Language Processing (NLP). Some of the currently working chatbots are:

- **Watson Assistant – IBM**

IBM Watson Assistant is a white name cloud administration that permits undertaking level programming designers to install a AI Virtual Assistant (VA) in the product they are creating and brand the associate as their own. The administration, which gives buyers access to Watson AI , is conveyed through the IBM Cloud. Watson Assistant, which utilizes Watson AI (ML) and Natural Language Understanding (NLU), is promoted to organizations that need to have the choice of keeping the information those courses through their remote helper private. Not at all like different merchants that total the data their remote helpers accumulate, has IBM offered engineers the decision of confining the data their associate assembles in a private cloud to secure exclusive bits of knowledge increased through client collaboration. Clients can communicate with the designer's application through an assortment of interfaces, including voice and contact.

- **Erika – Bank of America**

Erica is AI-driven and joins prescient investigation and normal language to help BofA portable application clients get to adjust data, exchange cash between records, send cash with Zelle, and calendar gatherings at monetary focuses. Clients can collaborate with Erica in any capacity they pick, including voice directions, messaging, or tapping choices on their telephone's screen. The more clients connect with Erica, the more the bot learns, and the better she moves toward becoming at giving assistance. Erica's learning of banking and budgetary administrations increments with each customer association. Erica likewise coordinates with Bank of America's spending and planning apparatuses and interfaces with client administration operators' consoles in call focuses. This implies when clients are utilizing Erica they can reach a genuine human operator with only a straightforward tap. Since clients are now confirmed while utilizing the BofA application, they don't need to re-recognize themselves to the operator. Also, the operators themselves can perceive what the client has been doing and getting some information about in Erica, so clients don't need to re-clarify what they're experiencing difficulty with.

- **H&M Chatbot**

H&M uses the H&M Kik bot as a brand extension tailored towards customizing the user's shopping experience. Whether consumers are looking for outfit inspiration or are debating trying out a new style, they can browse potential options by chatting with H&M on Kik as if they had a personal stylist at their fingertips. To start, the chatbot asks a few questions about the user's style by presenting two photos and asking users to simply pick "1" or "2." Thankfully, the exchange feels more like an entertaining game than a hassle. Then the user is given the choice to classify their style: Casual, Boho, Preppy, Classic, or Grunge. The H&M Kik bot then allows you to keep browsing similarly styled outfits featuring legwear, tops, shoes, and accent pieces, or start a new search altogether.

IV. Proposed Approach

Our chief review blog was by Andrej Karpathy. He talks in depth about the recurrent neural networks and how useful they in a variety of applications. He points out the differences between it and other Vanilla Neural Networks. He points out that vanilla neural networks' API is too constrained: they accept a fixed-sized vector as input (e.g. an image) and produce a fixed-sized vector as output (e.g. probabilities of different classes). Not only that: These models perform this mapping using a fixed amount of computational steps (e.g. the number of layers in the model). The core reason that recurrent nets are more exciting is that they allow us to operate over *sequences* of vectors: Sequences in the input, the output, or in the most general case both. He further on explains how they can be used for Character – Level Language Model which is very useful for our chatbot.

Natural Language Processing (NLP) is worried about how innovation can genuinely decipher and follow up on human language inputs. NLP permits innovation, for example, Amazon's Alexa to comprehend what you're stating and how to respond to it. Without NLP, AI that requires language inputs is generally futile. Like the past model with Amazon's Alexa, chatbots would almost certainly give practically no an incentive without Natural

Language Processing (NLP). Natural Language Processing is the thing that permits chatbots to comprehend your messages and react fittingly. When you communicate something specific with "Hi", the NLP lets the chatbot realize that you've posted a standard welcome, which thusly permits the chatbot to use its AI capacities to think of a fitting reaction. For this situation, the chatbot will probably react with an arrival welcoming. Without Natural Language Processing, a chatbot can't genuinely separate between the reactions "Hi" and "Farewell". To a chatbot without NLP, "Hi" and "Farewell" will both be just content based client inputs. Natural Language Processing (NLP) gives setting and significance to content based client inputs so AI can concoct the best reaction.

Algorithm: Sequence2Sequence model using LSTM Recurrent Neural Network Model

RECURRENT NEURAL NETWORK MODEL

Recurrent Networks take as their info the present information model they see, yet additionally what they have seen beforehand in time. The choice a recurrent net came to at time step t-1 influences the choice it will achieve one minute later at time step t. So recurrent networks have two wellsprings of info, the present and the ongoing past, which consolidate to decide how they react to new information, much as we do throughout everyday life.
Recurrent networks are recognized from feed forward networks by that criticism circle associated with their past choices, ingesting their own yields a great many moments as information. It is frequently said that recurrent networks have memory. Adding memory to neural networks has a reason: There is information in the arrangement itself, and recurrent nets use it to perform assignments that feed forward networks can't. That sequential information is protected in the recurrent network's hidden state, which figures out how to traverse many time ventures as it falls forward to influence the preparing of each new model. It is discovering connections between occasions isolated by numerous minutes, and these relationships are classified "long haul conditions", on the grounds that an occasion downstream in time relies on, and is an element of, at least one occasions that preceded.

We'll depict the way toward conveying memory forward scientifically:

The hidden state at time step t is h_t. It is a component of the contribution in the meantime step x_t, changed by a weight framework W (like the one we utilized for feedforward nets) added to the hidden state of the past time step h_t-1 increased by its very own hidden-state-to-hidden-state grid U, also called a progress lattice and like a Markov chain. The weight grids are channels that decide how much significance to accord to both the present information and the past hidden state. The mistake they create will return through backpropagation and be utilized to alter their loads until blunder can't go any lower.

The total of the weight input and hidden state is squashed by the capacity φ – either a strategic sigmoid capacity or tanh, depending – which is a standard instrument for gathering huge or little qualities into a calculated space, just as making inclinations serviceable for backpropagation. Since this criticism circle happens at each time venture in the arrangement, each hidden state contains follows of the past hidden state, yet in addition of every one of those that went before h_t-1 for whatever length of time that memory can continue. Given a progression of letters, a recurrent network will utilize the primary character to help decide its impression of the second character, with the end goal that an underlying q may lead it to deduce that the following letter will be u, while an underlying t may lead it to surmise that the following letter will be h.

DATASET : Cornell- Movie Corpus Dataset

This dataset contains a large metadata-rich collection of fictional conversations extracted from raw movie scripts:

- 220,579 conversational exchanges between 10,292 pairs of movie characters
- involves 9,035 characters from 617 movies
- in total 304, 713 utterances

The main reason to choose this dataset is because we wanted to implement a general chatbot that can have general conversation with the user like a friend instead of a specified chatbot used for some specific purpose, like a calendar assistant or navigation assistant.

The dataset consists of more than six hundred movies with thousands of conversation between tons of characters. The dataset consists of lot of metadata like genres, release date, and ratings which is descriptional data that has no point in being used for training. Only the movie-conversation and movie- lines data is going to be used for training of Chatbot. The movie-lines consists of a line spoken by a character in a movie with the respective line id. The movie-conversation consist of line ids that were involved in a particular conversation.

LIBRARIES USED:

In this model we made use of four libraries,

- *Numpy Library:* To work with arrays.
- *Tensor Flow:* To do the deep learning implementations.
- *Regular Expressions Library:* To clean the text and make them as simple as possible to learn the best conditions.
- *Time Library:* To measure training time of each epoch, epoch can be defined as a hyperparameter

that defines the number of time that the learning algorithm will work through the entire training dataset.

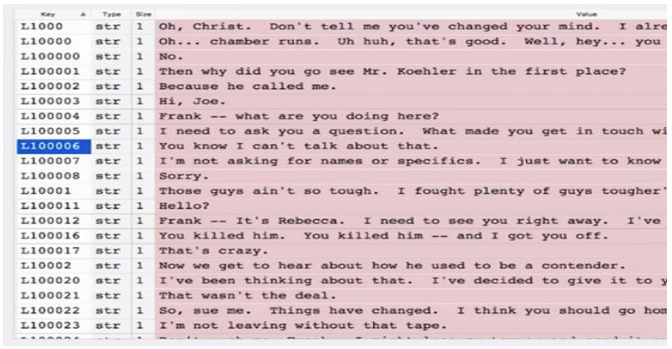
V. Data Preprocessing

Whenever an AI is built or whenever a machinery model is built it is very important to make the dataset compatible with the model that is going to be implemented. A neural network based chatbot model is going to be implemented and therefore the data will have to have special formats for the input. The main objective of data preprocessing for Chatbot is to get two different columns of data, one for questions(to be asked by the user) and another is answers(to be answered by the chatbot).

Initially we start by loading the dataset to our python file. Then we try to change the data in the dataset to the format required by our chatbot model.

- *Creating a dictionary that maps each lines and it's ids-*

This is done so that the dataset can contain two columns, an input and output because the inputs will be fed into the neural network and the outputs will be the target that is the real reply replied by the characters to the answers replied by the chatbot and that's how the chat bot will learn to speak whether it's prediction is close to the target or not.

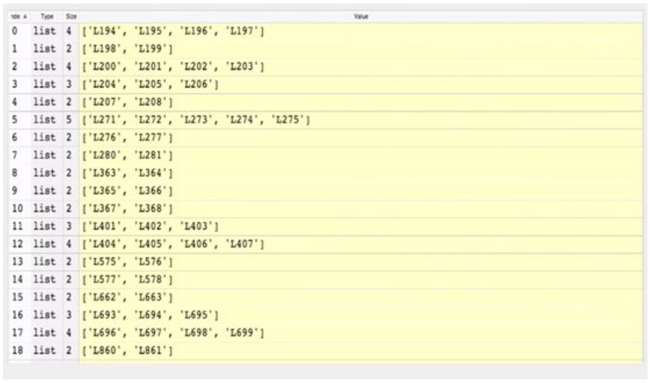


```
Key ▲    Type Size Value
L1000    str  1    Oh, Christ.  Don't tell me you've changed your mind.  I alre
L10000   str  1    Oh... chamber runs.  Uh huh, that's good.  Well, hey... you
L100000  str  1    No.
L100001  str  1    Then why did you go see Mr. Koehler in the first place?
L100002  str  1    Because he called me.
L100003  str  1    Hi, Joe.
L100004  str  1    Frank -- what are you doing here?
L100005  str  1    I need to ask you a question.  What made you get in touch wi
L100006  str  1    You know I can't talk about that.
L100007  str  1    I'm not asking for names or specifics.  I just want to know
L100008  str  1    Sorry.
L10001   str  1    Those guys ain't so tough.  I fought plenty of guys tougher'
L100011  str  1    Hello?
L100012  str  1    Frank -- It's Rebecca.  I need to see you right away.  I've
L100016  str  1    You killed him.  You killed him -- and I got you off.
L100017  str  1    That's crazy.
L10002   str  1    Now we get to hear about how he used to be a contender.
L100020  str  1    I've been thinking about that.  I've decided to give it to y
L100021  str  1    That wasn't the deal.
L100022  str  1    So, sue me.  Things have changed.  I think you should go hom
L100023  str  1    I'm not leaving without that tape.
```

A dictionary that consists of lines mapped to their respective ids.

- *Creating a list of all of the conversations-*

This is done because the conversation file consists of lot of metadata which is not required for training so only the line ids are filtered out and inserted into a list.

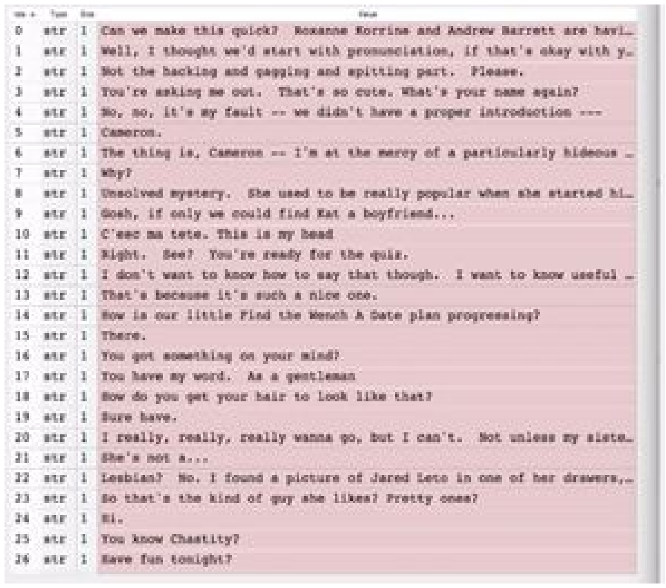


```
Index ▲ Type Size Value
0  list 4 ['L194', 'L195', 'L196', 'L197']
1  list 2 ['L198', 'L199']
2  list 4 ['L200', 'L201', 'L202', 'L203']
3  list 3 ['L204', 'L205', 'L206']
4  list 2 ['L207', 'L208']
5  list 5 ['L271', 'L272', 'L273', 'L274', 'L275']
6  list 2 ['L276', 'L277']
7  list 2 ['L280', 'L281']
8  list 2 ['L363', 'L364']
9  list 2 ['L365', 'L366']
10 list 2 ['L367', 'L368']
11 list 3 ['L401', 'L402', 'L403']
12 list 4 ['L404', 'L405', 'L406', 'L407']
13 list 2 ['L575', 'L576']
14 list 2 ['L577', 'L578']
15 list 2 ['L662', 'L663']
16 list 3 ['L693', 'L694', 'L695']
17 list 4 ['L696', 'L697', 'L698', 'L699']
18 list 2 ['L860', 'L861']
```

A list that consists of only lines involved in their respective conversations.

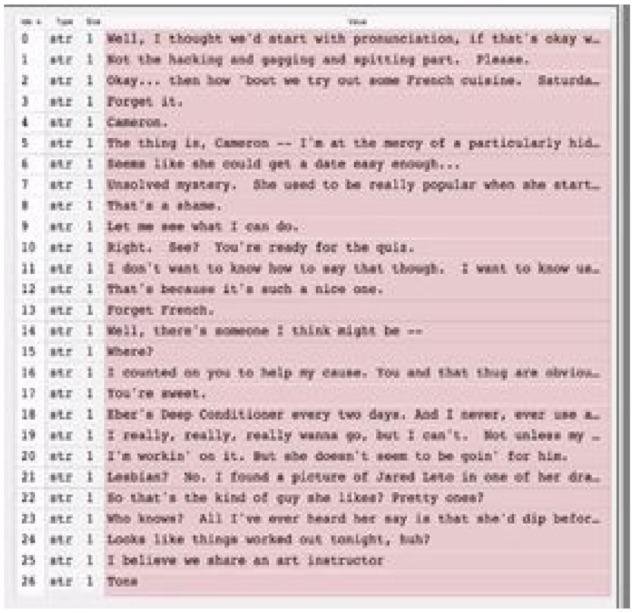
- *Getting separately the questions and the answers.*

The main purpose of this is to get two different lists questions(the inputs) and the answers(the target). From the above list each sublist first element is the question and the next elements are the answers.



```
Index ▲ Type Size Value
0  str 1 Can we make this quick?  Roxanne Korrine and Andrew Barrett are havi…
1  str 1 Well, I thought we'd start with pronunciation, if that's okay with y…
2  str 1 Not the hacking and gagging and spitting part.  Please.
3  str 1 You're asking me out.  That's so cute. What's your name again?
4  str 1 No, no, it's my fault -- we didn't have a proper introduction ---
5  str 1 Cameron.
6  str 1 The thing is, Cameron -- I'm at the mercy of a particularly hideous …
7  str 1 Why?
8  str 1 Unsolved mystery.  She used to be really popular when she started hi…
9  str 1 Gosh, if only we could find Kat a boyfriend...
10 str 1 C'esc ma tete. This is my head
11 str 1 Right.  See?  You're ready for the quiz.
12 str 1 I don't want to know how to say that though.  I want to know useful …
13 str 1 That's because it's such a nice one.
14 str 1 How is our little Find the Wench A Date plan progressing?
15 str 1 There.
16 str 1 You got something on your mind?
17 str 1 You have my word.  As a gentleman
18 str 1 How do you get your hair to look like that?
19 str 1 Sure have.
20 str 1 I really, really, really wanna go, but I can't.  Not unless my siste…
21 str 1 She's not a...
22 str 1 Lesbian?  No. I found a picture of Jared Leto in one of her drawers,…
23 str 1 So that's the kind of guy she likes? Pretty ones?
24 str 1 Hi.
25 str 1 You know Chastity?
26 str 1 Have fun tonight?
```

A list of questions



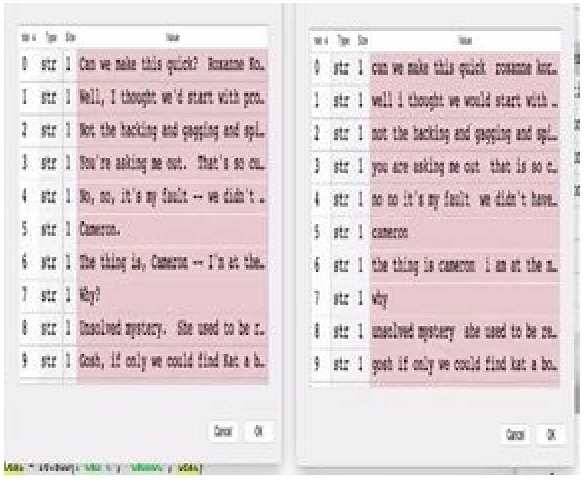
```
Index ▲ Type Size Value
0  str 1 Well, I thought we'd start with pronunciation, if that's okay w…
1  str 1 Not the hacking and gagging and spitting part.  Please.
2  str 1 Okay... then how 'bout we try out some French cuisine.  Saturda…
3  str 1 Forget it.
4  str 1 Cameron.
5  str 1 The thing is, Cameron -- I'm at the mercy of a particularly hid…
6  str 1 Seems like she could get a date easy enough...
7  str 1 Unsolved mystery.  She used to be really popular when she start…
8  str 1 That's a shame.
9  str 1 Let me see what I can do.
10 str 1 Right.  See?  You're ready for the quiz.
11 str 1 I don't want to know how to say that though.  I want to know us…
12 str 1 That's because it's such a nice one.
13 str 1 Forget French.
14 str 1 Well, there's someone I think might be --
15 str 1 Where?
16 str 1 I counted on you to help my cause. You and that thug are obviou…
17 str 1 You're sweet.
18 str 1 Eber's Deep Conditioner every two days. And I never, ever use a…
19 str 1 I really, really, really wanna go, but I can't.  Not unless my …
20 str 1 I'm workin' on it. But she doesn't seem to be goin' for him.
21 str 1 Lesbian?  No. I found a picture of Jared Leto in one of her dra…
22 str 1 So that's the kind of guy she likes? Pretty ones?
23 str 1 Who knows?  All I've ever heard her say is that she'd dip befor…
24 str 1 Looks like things worked out tonight, huh?
25 str 1 I believe we share an art instructor
26 str 1 Tons
```

A list of answers

Now, as we have got our input and target values we start cleaning the data.

- *Doing a first cleaning of the texts -*

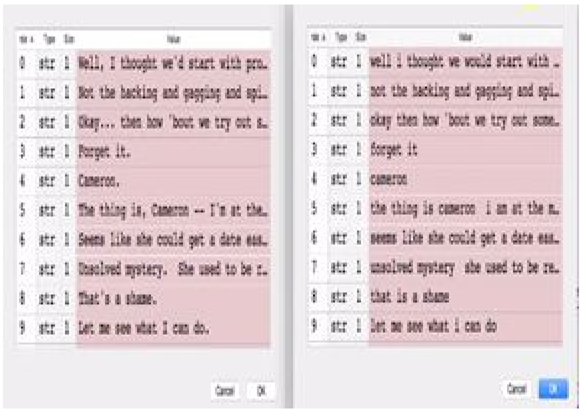
Everything in lower case, then all the apostrophe's will be removed, for example that's the same as that is when the chatbot is trained, and then all the special characters will be removed.



Index ▲	Type	Size	Value
0	str	1	Can we make this quick? Roxanne Ko...
1	str	1	Well, I thought we'd start with pro...
2	str	1	Not the hacking and gagging and spi...
3	str	1	You're asking me out. That's so cu...
4	str	1	No, no, it's my fault -- we didn't ...
5	str	1	Cameron.
6	str	1	The thing is, Cameron -- I'm at the...
7	str	1	Why?
8	str	1	Unsolved mystery. She used to be r...
9	str	1	Gosh, if only we could find Kat a b...

Index ▲	Type	Size	Value
0	str	1	can we make this quick roxanne kor...
1	str	1	well i thought we would start with ...
2	str	1	not the hacking and gagging and spi...
3	str	1	you are asking me out that is so c...
4	str	1	no no it's my fault we didn't have...
5	str	1	cameron
6	str	1	the thing is cameron i am at the m...
7	str	1	why
8	str	1	unsolved mystery she used to be re...
9	str	1	gosh if only we could find kat a bo...

Questions list after doing first cleaning



Index ▲	Type	Size	Value
0	str	1	Well, I thought we'd start with pro...
1	str	1	Not the hacking and gagging and spi...
2	str	1	Okay... then how 'bout we try out s...
3	str	1	Forget it.
4	str	1	Cameron.
5	str	1	The thing is, Cameron -- I'm at the...
6	str	1	Seems like she could get a date eas...
7	str	1	Unsolved mystery. She used to be r...
8	str	1	That's a shame.
9	str	1	Let me see what I can do.

Index ▲	Type	Size	Value
0	str	1	well i thought we would start with ...
1	str	1	not the hacking and gagging and spi...
2	str	1	okay then how 'bout we try out some...
3	str	1	forget it
4	str	1	cameron
5	str	1	the thing is cameron i am at the m...
6	str	1	seems like she could get a date eas...
7	str	1	unsolved mystery she used to be re...
8	str	1	that is a shame
9	str	1	let me see what i can do

Answers list after doing first cleaning

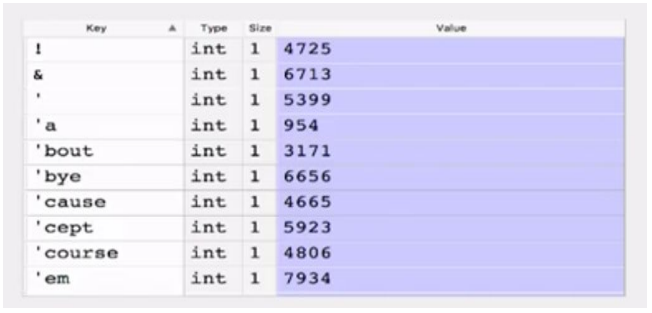
Then we perform more cleaning by removing the non-important words, i.e. the words that appear from time to time less than 5% actually of the whole corpus will be removed.

- *Creating a dictionary that maps each word to its number of occurrences-*

We first need to get all the words separately and for each of these words we get the number of occurrences, i.e. the number of times they appear in the corpus movie dialogues.

- *Creating two dictionaries that map the questions words and the answer words to a unique integer-*

Two essential tasks of Natural Language Processing required to be performed during data pre-processing, Tokenization and filtering non-frequent words. A threshold is set such that if the number of occurrences of a word is higher than the threshold the world is added to dictionary orelse the word is not included. Doing these 5% of the least appearing words are removed.



Key ▲	Type	Size	Value
!	int	1	4725
&	int	1	6713
'	int	1	5399
'a	int	1	954
'bout	int	1	3171
'bye	int	1	6656
'cause	int	1	4665
'cept	int	1	5923
'course	int	1	4806
'em	int	1	7934

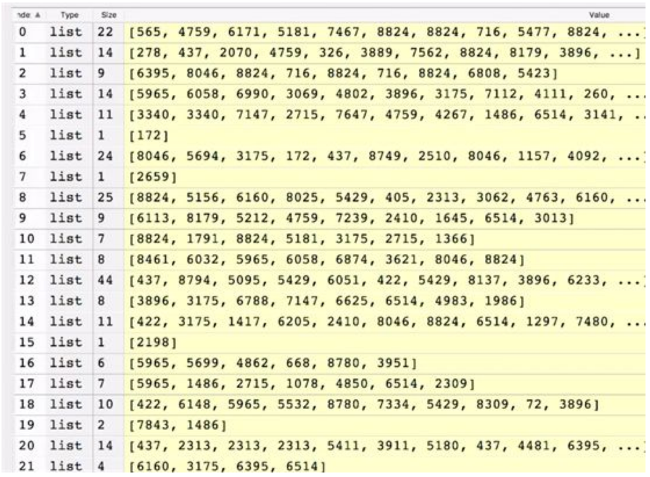
Mapping of words to a unique integer

Further, the last tokens are added to these dictionaries. The last tokens are useful for the encoder and the decoder in the Seq2Seq model which are the start of the string that we're going to encode by as the end of string we going to encode by EOS, it is very important for Chatbot that the process training data and the sequences in the batch should all have the same length and therefore we have to input this toke in an empty position. There are four tokens,

1. **PAD** - The input
2. **EOS** - End Of String
3. **OUT** - Calls out all the words that were filtered out by the last dictionary.
4. **SOS** - Start of string, the first token that should start in the decoding layer.

- *Translating all the questions and answers into integers*

For the above dictionary the inverse dictionary is created because we will need inverse mapping from the integers to the answers words in the implementation of the Seq2Seq models. Then the end of string token is added to the end of every answer. Now, all the questions and answers are translated into integers by replacing all the words that were filtered out by <OUT>.

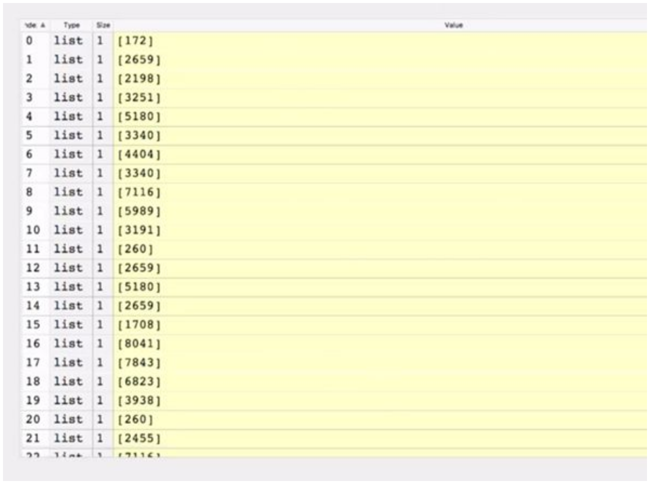


ndex ▲	Type	Size	Value
0	list	22	[565, 4759, 6171, 5181, 7467, 8824, 8824, 716, 5477, 8824, ...
1	list	14	[278, 437, 2070, 4759, 326, 3889, 7562, 8824, 8179, 3896, ...]
2	list	9	[6395, 8046, 8824, 716, 8824, 716, 8824, 6808, 5423]
3	list	14	[5965, 6058, 6990, 3069, 4802, 3896, 3175, 7112, 4111, 260, ...
4	list	11	[3340, 3340, 7147, 2715, 7647, 4759, 4267, 1486, 6514, 3141, ..
5	list	1	[172]
6	list	24	[8046, 5694, 3175, 172, 437, 8749, 2510, 8046, 1157, 4092, ...
7	list	1	[2659]
8	list	25	[8824, 5156, 6160, 8025, 5429, 405, 2313, 3062, 4763, 6160, ...
9	list	9	[6113, 8179, 5212, 4759, 7239, 2410, 1645, 6514, 3013]
10	list	7	[8824, 1791, 8824, 5181, 3175, 2715, 1366]
11	list	8	[8461, 6032, 5965, 6058, 6874, 3621, 8046, 8824]
12	list	44	[437, 8794, 5095, 5429, 6051, 422, 5429, 8137, 3896, 6233, ...
13	list	8	[3896, 3175, 6788, 7147, 6625, 6514, 4983, 1986]
14	list	11	[422, 3175, 1417, 6205, 2410, 8046, 8824, 6514, 1297, 7480, ...
15	list	1	[2198]
16	list	6	[5965, 5699, 4862, 668, 8780, 3951]
17	list	7	[5965, 1486, 2715, 1078, 4850, 6514, 2309]
18	list	10	[422, 6148, 5965, 5532, 8780, 7334, 5429, 8309, 72, 3896]
19	list	2	[7843, 1486]
20	list	14	[437, 2313, 2313, 2313, 5411, 3911, 5180, 437, 4481, 6395, ...
21	list	4	[6160, 3175, 6395, 6514]

Words in questions translated into integers

- *Sorting questions and answers by the length of questions-*

Sorting the questions and answers by the length of questions is very important as it will speed up the training, the sorted list helps the chatbot to train on smaller data first and then get better on large data. It also helps to reduce the loss and also reduces padding during the training.



de ▲	Type	Size	Value
0	list	1	[172]
1	list	1	[2659]
2	list	1	[2198]
3	list	1	[3251]
4	list	1	[5180]
5	list	1	[3340]
6	list	1	[4404]
7	list	1	[3340]
8	list	1	[7116]
9	list	1	[5989]
10	list	1	[3191]
11	list	1	[260]
12	list	1	[2659]
13	list	1	[5180]
14	list	1	[2659]
15	list	1	[1708]
16	list	1	[8041]
17	list	1	[7843]
18	list	1	[6823]
19	list	1	[3938]
20	list	1	[260]
21	list	1	[2455]
22	list	1	[7116]

Sorted list based on the length

VI. Implementation

We implemented sequence to sequence model to create a chat bot. the sequential to sequential model has two bi-directional recurrent neural networks with many LSTM units connected. These two RNN's are kept in sequential way such that the output of the first one becomes the input to the second one. The first one is called the Encoder and the second one is called the Decoder.

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. Since chatbot requires sequences of inputs mapped to sequences of outputs, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector

In when we do not know the length of the input vector and the output the model gives it pose a challenge for Deep neural networks because they need that the dimensionality of the inputs and outputs is already fixed. In this paper, we show that a straightforward application of the Long Short-Term Memory (LSTM) architecture [1] can solve the problem of variable length input and out that is general sequence to sequence problems. The idea is to use one LSTM(Encoder) to read the input sequence, one time step at a time, to obtain large fixed dimensional vector representation, and then to use another LSTM(decoder) to get the output sequence from that vector (fig. 1). The second LSTM is essentially a RNN model [3, 2, 4] except that it is dependent on the input sequence that is the encoded vector in the encoding part. The LSTM's ability to successfully learn on data with long range temporal dependencies makes it a intuitive choice for this application due to the considerable time lag between the inputs and their corresponding outputs (fig. 1).

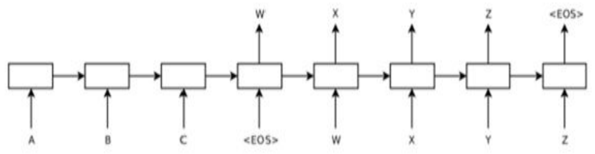
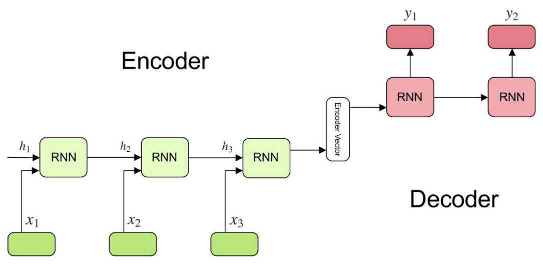


Figure 1: Our model reads an input sentence "hi how are you" and produces "i am good" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short-term dependencies in the data that make the optimization problem much easier.

MODEL:



Encoder-decoder sequence to sequence model

The RNN[5,6] is a modification over artificial feed forward neural network. Given a sequence of inputs (x1, . . . , xT), a standard RNN computes a sequence of outputs (y1, . . . , yT) by iterating the following equation.

$$\begin{aligned} h_t &= \operatorname{sigm}\left(W^{\mathrm{hx}} x_t + W^{\mathrm{hh}} h_{t-1}\right) \\ y_t &= W^{\mathrm{yh}} h_t \end{aligned}$$

The RNN can easily map sequences to sequences that is the sequence of inputs to the chatbot and the sequence the chatbot should ideally produce. whenever the alignment between the inputs the outputs is known ahead of time. However, it is not clear how to apply an RNN to problems whose input and the output sequences have variable lengths with complicated and non-monotonic relationships. The simplest strategy for general sequence learning is to map the input sequence to a fixed-sized vector using one RNN, and then to map the vector to the target sequence with another RNN (this approach has also been taken by Cho et al. [7]). While it could work in principle since the RNN is provided with all the relevant information, it would be difficult to train the RNNs due to the resulting long-term dependencies (figure 1) . However, the Long Short-Term Memory (LSTM) [1] is known to learn problems with long range temporal dependencies, so an LSTM may succeed in this setting. The goal of the LSTM is to estimate the conditional probability p(y1, . . . , yT′ |x1, . . . , xT) where (x1, . . . , xT) is an input sequence and y1, . . . , yT′ is its corresponding output sequence whose length T ′ may differ from T . The LSTM computes this conditional probability by first obtaining the fixed dimensional representation v of the input sequence (x1, . . . , xT) given by the last hidden state of the LSTM, and then computing the probability of y1, . . . , yT′ with a standard LSTM-LM formulation whose initial hidden state is set to the representation v of x1, . . . , xT :

$$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

In this equation, each p(yt|v, y1, . . . , yt−1) distribution is represented with a SoftMax over all the words in the vocabulary. We use the LSTM formulation from Graves. Note that we require that each sentence ends with a special end-of-sentence symbol "EOS", which enables the model to define a distribution over sequences of all possible lengths. The overall scheme is outlined in figure 1, where the shown LSTM computes the representation of "A", "B", "C", "" and then uses this representation to compute the probability of "W", "X", "Y", "Z", "". Our actual models differ from the above description in three important ways. First, we used two different LSTMs: one for the input sequence and another for the output sequence, because doing so increases the number model parameters at negligible computational cost and makes it natural to train the LSTM on multiple language pairs simultaneously. Second, we found that deep LSTMs significantly outperformed shallow LSTMs, so we chose an LSTM with four layers. Third, we found it extremely valuable to reverse the order of the words of the input sentence. So for example, instead of mapping the sentence a, b, c to the sentence α, β, γ, the LSTM is asked to map c, b, a to α, β, γ, where α, β, γ is the translation of a, b, c. This way, a is in close proximity to α, b is fairly close to β, and so on, a fact that makes it easy for SGD to "establish communication" between the input and the output. We found this simple data transformation to greatly improve the performance of the LSTM.

DECODING and RESCORING:

The core of our experiments involved training a large deep LSTM on many sentence pairs. We trained it by maximizing the log probability of a correct answer T given the source sentence S, so the training objective is

$$1/|S| \sum_{(T,S) \in S} \log p(T|S)$$

where S is the training set. Once training is complete, we produce translations by finding the most likely translation according to the LSTM:

$$\hat{T} = \arg\max_{T} p(T|S)$$

we used bean search decoding instead of using a greedy search. We take the answer with high probability. which maintains a small number B of partial hypotheses, where a partial hypothesis is a prefix of some translation. At every timestep we extend each partial hypothesis in the beam with every possible word in the vocabulary. This greatly increases the number of the hypotheses, so we discard all but the B most likely hypotheses according to the model's log probability. As soon as the "EOS" symbol is appended to a hypothesis, it is removed from the beam and is added to the set of complete hypotheses. While this decoder is approximate, it is simple to implement. Interestingly, our system performs well even with a beam size of 1, and a beam of size 2 provides most of the benefits of beam search (Table 1). We also used the LSTM to rescore the 1000-best lists produced by the baseline system [29]. To rescore an n-best list, we computed the log probability of every hypothesis with our LSTM and took an even average with their score and the LSTM's score.

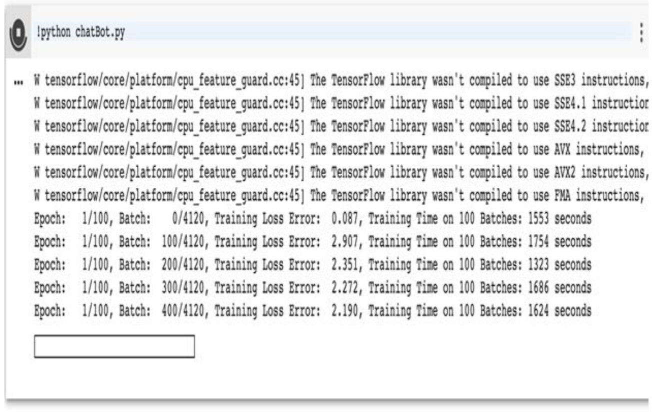
After the data set is preprocessed and we get that structure of questions and answers from the Cornell's movie corpus. Now we take each pair of question and answer to train. Each word of the question is converted into the one hot vector based on the vocabulary created in the preprocessing step. This one hot encode vector in multiplied by 512-dimensional word embeddings matrix. The resultant fixed-size matrix which represents the word is fed into the model.

We initialized all of the LSTM's parameters with the uniform distribution between -0.08 and 0.08. stochastic gradient descent without momentum, with a fixed learning rate alpha of 0.1 is used. the training is done for a total of 10 epochs. We used batches of sequences for the gradient and divided it the size of the batch (namely, 128). we found that even though there was not much problem of vanishing gradient, the exploding gradient problem was there. Thus, we used break out to deactivate some neurons[10, 25] by scaling it when its norm exceeded a threshold. For each training batch, we compute s = kgk2 , where g is the gradient divided by 128. If s > 5, we set g = 5g s .

VII. Experimental Results and Discussions

Leveraging the power of GPU processing, we ran our model in google collab GPU runtime environment for 10 epochs and saved the model state. the conversation shown below is the conversation having between the user and the model which was run for 10 epochs in google collab GPU runtime.

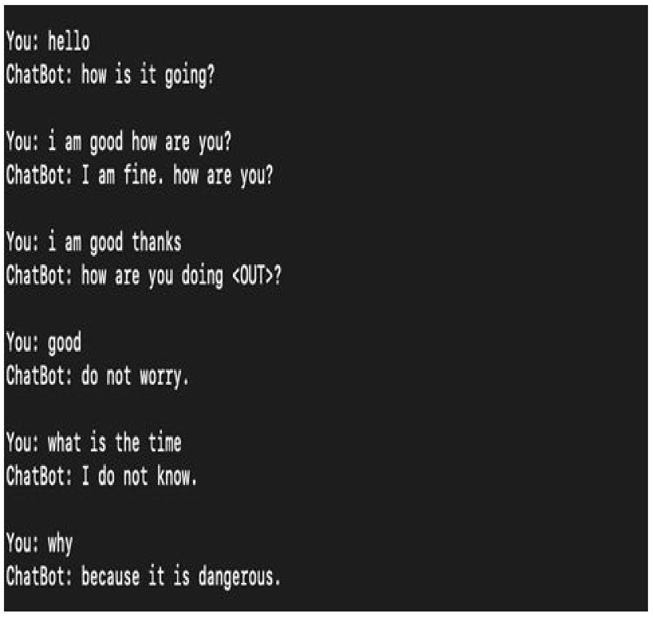
Already after 10 epochs our model learns some context about the question and answers accordingly. with just cpu processing training our model will take weeks. after our gpu free time was over we ran our model in google collab cpu environment, below is the picture which is showing our model running in cpu runtime of google collab and you can see the training loss error after each batch in the first epoch.



```
!python chatBot.py

... W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE3 instructions,
    W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructior
    W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 instructior
    W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX instructions,
    W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX2 instructions,
    W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use FMA instructions,
    Epoch:   1/100, Batch:    0/4120, Training Loss Error:  0.087, Training Time on 100 Batches: 1553 seconds
    Epoch:   1/100, Batch:  100/4120, Training Loss Error:  2.907, Training Time on 100 Batches: 1754 seconds
    Epoch:   1/100, Batch:  200/4120, Training Loss Error:  2.351, Training Time on 100 Batches: 1323 seconds
    Epoch:   1/100, Batch:  300/4120, Training Loss Error:  2.272, Training Time on 100 Batches: 1686 seconds
    Epoch:   1/100, Batch:  400/4120, Training Loss Error:  2.190, Training Time on 100 Batches: 1624 seconds
```

Below is a sample of how our chatbot model after trained in GPU runtime of google collab, replying to the questions asked by the user. It was not giving exceptionally appropriate answers but with more training it will give better results.



```
You: hello
ChatBot: how is it going?

You: i am good how are you?
ChatBot: I am fine. how are you?

You: i am good thanks
ChatBot: how are you doing <OUT>?

You: good
ChatBot: do not worry.

You: what is the time
ChatBot: I do not know.

You: why
ChatBot: because it is dangerous.
```

VIII. Conclusion and Future Work

We want to conclude by saying that we have created a satisfactory chat bot which can make generalized conversations with the users. given more training to the model and more gpu processing we can improve the results of the model.

Most importantly, we demonstrated that a simple, straightforward and a relatively unoptimized approach can be used to develop a chat bot, so further work will likely lead to even greater conversation accuracies. These results suggest that our approach will likely do well on other challenging sequence to sequence problems.

References

[1] S. Hochreiter and J. Schmidhuber, Long short-term memory Neural Computation, 1997.

[2] T. Mikolov, M. Karafi´at, L. Burget, J. Cernock`y, and S. Khudanpur. Recurrent neural network based language model. In INTERSPEECH, pages 1045–1048, 2010.

[3] D. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. Nature, 323(6088):533–536, 1986.

[4] M. Sundermeyer, R. Schluter, and H. Ney. LSTM neural networks for language modeling. In INTERSPEECH, 2010

[5] P. Werbos. Backpropagation through time: what it does and how to do it. Proceedings of IEEE, 1990.

[6] D. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. Nature, 323(6088):533–536, 1986.

[7] K. Cho, B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Arxiv preprint arXiv:1406.1078, 2014