# A Chatbot using Recurrent Neural Network

Aviral Mithal , Dhaval Vora , Dhruv Bhutani , Nikhil Tekwani*

*201301178 , 201301153 , 201301176 , 201301151*

E-mail:

## Introduction

This project intend to make a chatbot that converses with the user and according to the need of the user provides him with interesting facts and news about sports. The project is divides into two parts

1. Language modelling

2. Chat Bot models.

In Language modelling we try to make sensible meaningful sentences out of the data provided. In the next part i.e the chatbot model we try to figure out the context of the conversation and the post a meaningful reply based on that.

## Language Models

We use two language models.

- Markov Chains
- Recurrent Neural network

We then use these two in the chatbot models to generate meaningful replies.

## Markov Models

A Markov chain has no memory of previous states: the next state (word, in this case) is chosen based on a random dice roll and a lookup into a table of the states that tend to historically follow the current state in the input corpus. Given an adequate input corpus, this works almost uncannily well.

- Based on training data , form a Markov chain with all the probabilities of words following certain words.
- Start from a word and choose a following word based on the probabilities.
- repeat this process till end of line.

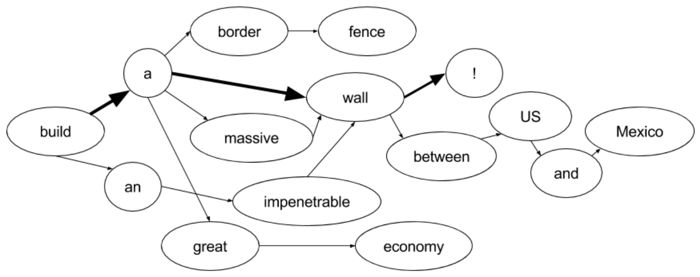


Figure 1: A markovian chain of Donald Trumps speeches

## Recurrent Neural Network

The idea behind RNNs is to make use of sequential information. In a traditional neural network it is assumed that all inputs (and outputs) are independent of each other. But for many tasks thats a very bad idea. If one wants to predict the next word in a sentence he should better know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a memory

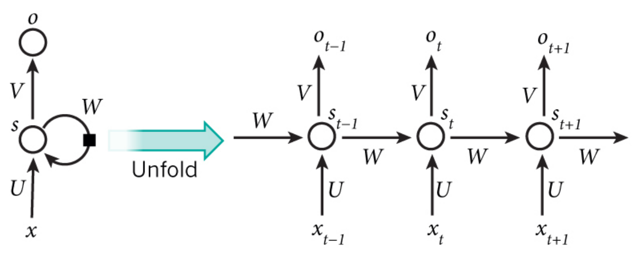


Figure 2: A recurrent neural network

which captures information about what has been calculated so far. In the figure 2, $x_t$ is the input at time step t and $o_t$ is the output at step t.

Basic RNNs take each element of a sequence, multiply the element by a matrix, and then sum the result with the previous output from the network. This is described by the following equation:

$$h_t = activation(X_t W_t + h_{t-1} W_t)$$

We try to optimise the error(cross entropy loss):

$$L(y,o) = \frac{1}{N} \sum_{n \in N} y_n log o_n$$

Unlike simple language models that just try to predict a probability for the next word given the current word, RNN models capture the entire context of the input sequence. Therefore, RNNs predict the probability of generating the next word based on the current word, as well as all previous words.

In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps. Consider trying to predict the last word in the text I grew up in France I speak fluent $French$. Recent information suggests that the next word is probably the name of a language, but if one wants to narrow down which language, the context of France is required, from further back. Its entirely possible for the gap between the relevant information and the point where it is needed to become very large.

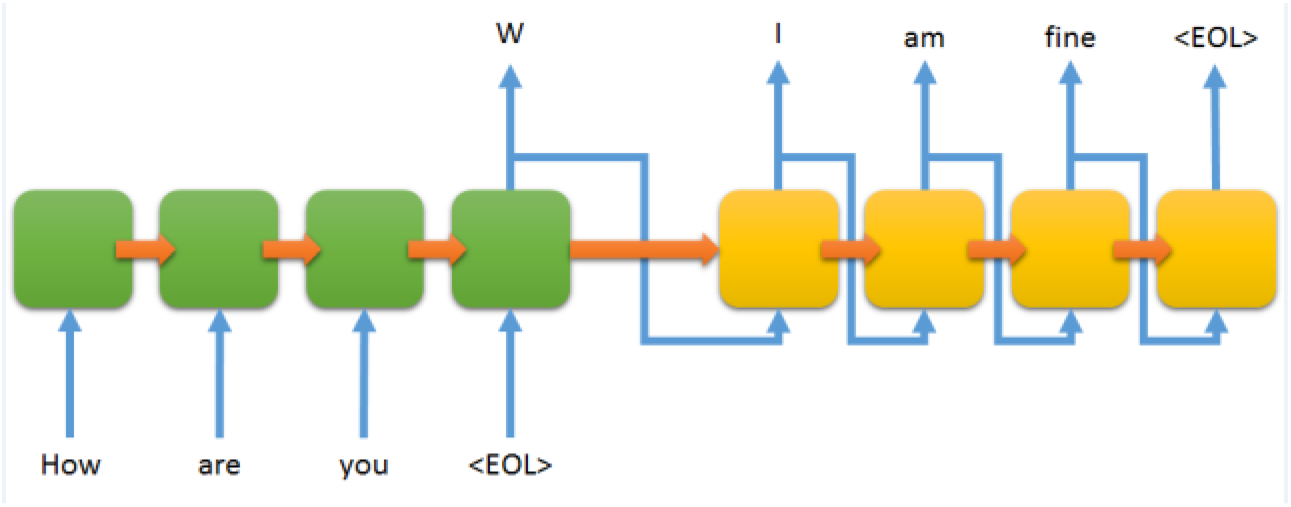
Unfortunately, if that gap grows very large, RNNs become unable to learn to connect the information.

## Chat Bot Models

### Sequence to sequence Models

A basic sequence-to-sequence model consists of two recurrent neural networks (RNNs): an encoder that processes the input and a decoder that generates the output.Sequence to sequence models build on top of language models by adding an encoder and a decoder step. In the encoder step, a model converts an input sequence (such as an English sentence) into a fixed representation. In the decoder step, a language model is trained on both the output sequence as well as the fixed representation from the encoder.

During training, the true output sequence is given to the model, so learning can be done by backpropagation. The model is trained to maximize the cross entropy of the correct sequence given its context



Scheme 1: Sequence to sequence model

## NLP techniques

Whenever the user posts a message , we find the topic(The main theme) of the message via NLP techniques.

We use Context free Grammar (The idea of basing a grammar on constituent structure) to generate a parse tree. In the resulting parse trees, find all nouns which are only subordinate to Noun-Phrase-like constituents.

This way we find the topic of a message posted by the user.

## Hybrid Model

We define the hybrid model in which we use both the markov chain and RNN model for generation of replies.

If the message is posted as a question with the topic present in the database for the Markov chains , the chatbot will post a reply based on the Markov model.

If no such topic exists or the message is not a question or multiple topics exist, we use the Sequence to Sequence model to generate a reply and post it.

The Chatbot keeps on adding data to the Markov chain by using the users chat logs with the bot.

## References

(1) Lifeng Shang, Zhengdong Lu, and Hang Li. *Neural Responding Machine for Short-Text Conversation.* ACL 2015.

(2) Ilya Sutskever James Martens Geoffrey Hinton. *Generating Text with Recurrent Neural Networks.*

(3) https://indico.io/blog/sequence-modeling-neural-networks-part2-attention-models/