***AI Chatbot with Machine Learning & NLP***

MD SAMI UL ISLAM

Computer Science & Engineering

BRAC University

Dhaka, Bangladesh

samiulislam93@hotmail.com

SHEAKH FAHIM AHMMED JOY

Computer Science & Engineering

BRAC University

Dhaka, Bangladesh

fahim229@gmail.com

*Abstract*—Chat-bots are one of the most used and powerful tool in the online sector. Online Shops, Restaurant facebook pages and numerous public figures installed this tool in their messenger accounts to avoid the hassle but still be responsive with clients to meet their demands. An intelligent software-agent can do this who understands human language. Natural Language Processing is the main role featuring in all AI Chat-bots. NLP makes a pattern of human words to help the agent understand and learn from experience. Machine Learning is another aspect which strengthens the bot and makes it more realistic to get more accurate and relevant answers.

Keywords—machine learning, natural language processing, tensorflow, softmax, bAbI, beam search.

# Introduction

This paper is going to show how to build a chat-bot in step by step process. First, we collected the dataset which more likely to serve our interest and goal. For an example, data that are simple but fetches accurate answer. Then, we created a model or selected a few algorithms to run the bot in that fashion. Next, we had to train the bot to understand the questionnaire and deliver answers which should be accurate and rational in and around 98%. Finally, we have test the chat-bot if it is working properly according to what it should be doing.

# Background study

## Using Corpora in Machine-Learning Chatbot Systems

A chat-bot is a machine conversation system which interacts with human users via natural conversational language. Software to machine-learn conversational patterns from a transcribed dialogue corpus has been used to generate a range of chat-bots speaking various languages and sublanguages. Corpora have been widely used by linguists to develop and refine “language models”, descriptions of lexis, grammar, dialogue, etc. Language models can also be automatically extracted or machine-learnt from corpora, to drive language analysis systems; for example, machine-learning of Part-of-Speech taggers from PoS-tagged corpora (Atwell 1983; Atwell et al. 2000a); machine-learning to automatically cluster words in a corpus into grammatical classes (Atwell & Drakos 1987; Hughes & Atwell 1994); machine-learnt grammar checkers (Atwell 1983, 1987); machine learning of parsers from parsed corpus treebanks (Atwell 1983, 1988, 1996; Atwell et al. 1984), adapting speech recognition models for English language learners (Atwell et al. 2000b, 2003).

## Building a Chatbot with Serverless Computing

Cloud providers are offering a growing number of cognitive services, such as machine learning, translation, and analytics, as building blocks for developing AI applications. One such application is a “chat-bot,” which uses a conversational interface to interact with the user. Chat-bots have been around since the 1960s, but recently there has been a rapid increase in the number of chat-bots due in part to a wide market adoption of mobile and “smart” devices. Chat-bots are now embedded in popular messaging programs and also appear as stand-alone services like Amazon Alexa, Microsoft’s Cortana, and Apple Siri. Serverless has recently emerged as an alternative way of creating backend applications. Serverless does not require dedicated infrastructure. Major cloud vendors such as Amazon Web Services, Google, Microsoft, and IBM have created versions of serverless. Serverless lets the developer deploy “functions” (serverless is also referred to as Functions-as-a-Service) into a shared platform that is maintained by the vendor. The functions are typically standard code snippets in popular languages that execute in a stateless fashion. It is the vendor’s responsibility to keep the infrastructure running smoothly and scale resources to satisfy function executions due to changes in user demand. Serverless life-cycle costs are typically lower than costs for dedicated infrastructure as serverless vendors do not charge for idle time.

## CLARA: A Multifunctional Virtual Agent for Conference Support and Touristic Information

There is an increasing interest in using conversational agents for both web and mobile applications since they allow users to quickly find corporate or product information while at the same time engage users by providing them with relevant notifications about new products, offering recommendations, or simply by being able to handle user complains or feedback (i.e. chat-bot capabilities).There are several examples of conversational agents used for different domains in the literature, such as health-care, weather forecast, tutoring, tourism, etc. Probably, the most popular applications are Apple ́s Siri, Google Now and Cortana. These systems are able to provide information for multiple tasks and domains including making appointments, sending text messages, providing weather or transportation information, or searching the web. However, up to the best of our knowledge, there is not any conversational agent specifically designed for conferences, which at the same time could handle local information about the place where the conference is held; which certainly is very useful for first time visitors.

.

# Data description

## Understanding Problem

## Using deep learning domain-specific chat bots can “learn” about the topic provided to it and then be able to answer questions related to it. The applications of a technology like this are endless. You just provide data about a topic and watch the bot become an expert at it.

In this paper, we will show how to build a blog that can answer logical reasoning questions. Yes it can be done!

## Data

The data used here was made open source by Facebook AI research. bAbI dataset was created by Facebook towards the goal of automatic text understanding and reasoning. It is a set of 20 QA tasks, each consisting of several context-question-answer triplets. Each task aims to test a unique aspect of reasoning and is, therefore, geared towards testing a specific capability of QA learning models.

An example from the second task, Two Supporting Facts (QA2), is below:

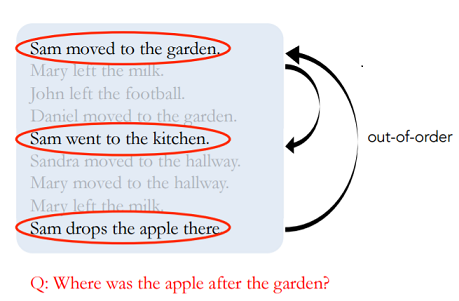
1. John moved to the bedroom.  
2. **Mary grabbed the football there.**  
3. Sandra journeyed to the bedroom.  
4. Sandra went back to the hallway.  
5. Mary moved to the garden.  
6. **Mary journeyed to the office.**  
Question: Where is the **football**?

Answer: office based on statements 2, 6

There are 1000 training examples for each question. The dataset has synthetically generated stories and hence the vocabulary is very limited and the sentence forms are very constrained. On the one hand, these limitations make bAbI an ideal dataset for learning — not much data cleansing needs to be done and one can focus on model building. On the other hand, they raise questions about the ability to generalize results on bAbI to QA in a less tightly controlled environment.

## Deep Learning Model

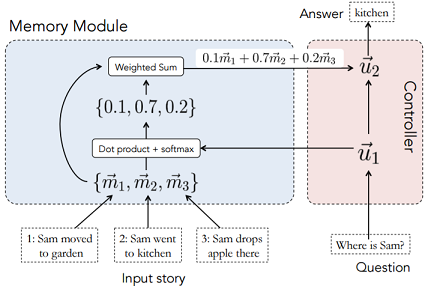
For this problem we use an end to end memory network which was designed by Facebook AI Research. Let’s first discuss why a recurrent neural network with LSTM cells won’t work well for this problem.

1. Data may have to be accessed out of order. See example below:
2. We need a long term memory if we want to network to “memorize” a book or a video and LSTM cells practically don’t work for long term memory problems.

The end to end memory networks use an external memory to solve this problem. And it can perform multiple look ups (hops) in the data to solve for out of order data problem. What does this mean in simple terms?

This model has two modules — a memory module and a controller module. The memory module has the story we want to tell written as vectors. Since computers cannot understand words, we need to convert words into numbers/vectors. Instead of just converting every word to a number, we use a technique called embedding which gives words with similar meanings similar vectors. For example since the words “pick up” or “lift” have similar meaning they will have similar vectors and it helps the machine better understand the context between words. Tensorflow embedding functionality can be used for generating word to vectors.

The controller module has the question that is being asked written as a vector. See example below:



We then do a dot product of the controller function and the memory vectors. And after that we do a softmax which scales the total probability to 1. And this is passed back to the controller. In the example above, since this neural network takes into account the order of sentences, it gives higher weigh-age to the second sentence. We can think of the resulting probabilities (0.1, 0.7, 0.2, above) as implying which vector is the network paying “attention” to.

This process can be repeated several times (also called hops) and the controller has the ability to learn and re look at the memory vectors and make a choice again. See below how the network focuses on different vectors during hops by building on what is has learned in the previous hop.

Finally, the input is passed to a decoder and the vectors and converted to words.

# training & testing

We implemented “Beam Search” as the algorithm. We trained the bot from dataset and learn by experience.

## Beam Search

## Beam search is a restricted or modified version of either a breadth-first search or a best-first search. It is restricted in the sense that the amount of memory available for storing the set of alternative search nodes is limited, and in the sense that non-promising nodes can be pruned at any step in the search (Zhang, 1999). The pruning of non-promising nodes is determined by problem-specific heuristics (Zhang, 1999). The set of most promising, or best alternative, search nodes is called the “beam” (Xu and Fern, 2007). Essentially, beam search is a forward-pruning, heuristic search.

## 

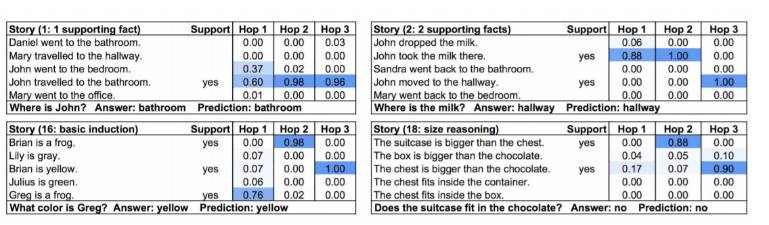
## Implementing Beam Search Decoder for NLP

Natural language processing tasks, such as caption generation and machine translation, involve generating sequences of words.

Models developed for these problems often operate by generating probability distributions across the vocabulary of output words and it is up to decoding algorithms to sample the probability distributions to generate the most likely sequences of words.

In this paper, we have discovered the beam search decoding algorithms that can be used on text generation problems.

In natural language processing tasks such as caption generation, text summarization, and machine translation, the prediction required is a sequence of words.

It is common for models developed for these types of problems to output a probability distribution over each word in the vocabulary for each word in the output sequence. It is then left to a decoder process to transform the probabilities into a final sequence of words.

You are likely to encounter this when working with recurrent neural networks on natural language processing tasks where text is generated as an output. The final layer in the neural network model has one neuron for each word in the output vocabulary and a softmax activation function is used to output a likelihood of each word in the vocabulary being the next word in the sequence.

Decoding the most likely output sequence involves searching through all the possible output sequences based on their likelihood. The size of the vocabulary is often tens or hundreds of thousands of words, or even millions of words. Therefore, the search problem is exponential in the length of the output sequence and is intractable (NP-complete) to search completely.

In practice, heuristic search methods are used to return one or more approximate or “good enough” decoded output sequences for a given prediction.

##### References

1. Bayan Abu Shawar, Eric Artwell, “Using Corpora in Machine-Learning Chatbot Systems”.
2. Mengting Yan, Paul Castro, Perry Cheng, Vatche Ishakian, “Building A Chat-bot with Serverless Computing”.
3. Luis Fernando D’Haro, Seokhwan Kim, Kheng Hui Yeo, Ridong Jiang, Andreea I. Niculescu, Rafael E. Banchs, Haizhou Li, “CLARA: A Multifunctional Virtual Agent for Conference Support and Touristic Information”.
4. https://towardsdatascience.com/building-an-ai-chat-bot-e3a05aa3e75f
5. https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/