International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 03 | Mar-2018 www.irjet.net p-ISSN: 2395-0072

Chatbot Using Gated End-to-End Memory Networks

Jincy Susan Thomas[1], Seena Thomas[2]

[1]Jincy Susan Thomas, M.Tech CSE, LBS Institute of Technology for Women, Trivandrum, Kerala, India
[2]Seena Thomas, Assistant Professor, Dept. of Computer Science, LBS Institute of Technology for Women, Trivandrum, Kerala, India

---

Abstract - *Intelligent bot systems interact with users in natural language. Existing traditional chatbot systems are rule based that stops scaling up to real world domains. While using deep learning methods, all dialogs in the chatbot system are trained from the dialogs themselves escapes the limitations of this. The paper proposes deep learning method gated end-to-end memory networks and we can show how chatbot systems can be used in real time applications. This model is learned in an end-to-end fashion without the use of any additional supervision. We set up the context of hospital reservation that generate and manipulate sentences for properly conducting conversations, issue api calls and retrieve responses based on api calls. We analyse the drawbacks of existing rule based system and propose modifications to those in the new dataset with hospital information that improve the responses. We can reveal that a dialogue system based on memory network is able to achieve good results on non-trivial operations.*

Key Words: intelligent, natural language, rule based, gated end-to-end memory networks, supervision, hospital, dialogue system

1. INTRODUCTION

Dialogue systems can be divided into goal-driven systems such as technical support services, and non-goal-driven systems, such as language learning tools or computer game characters. Traditional dialogue systems is slot-filling[12][13] which predefines the structure of a dialogue state as a set of slots to be filled during the dialog. An existing implementation is a restaurant reservation system, where the slots can be location, price range or type of cuisine of a restaurant. Slot-filling has proven to be reliable, but hard to scale to new domains. Slot filling uses rule based approach eg., siri. In a rule based approach, a bot answers question based on some rules on which it is trained on. The creation of these bots are relatively straightforward using some rule-based approach, but the bot is not efficient in answering questions, whose pattern does not match with the rules on which the bot is trained. Those bots can be created by using language like Artificial Intelligence Markup Language(AIML), a language based on XML that allow developer's write rules for the bot to follow. Another drawback is writing rules for different scenarios is very time consuming and it is impossible to write rules for every possible scenario. So these bots can handle simple queries but fail to manage complex queries.

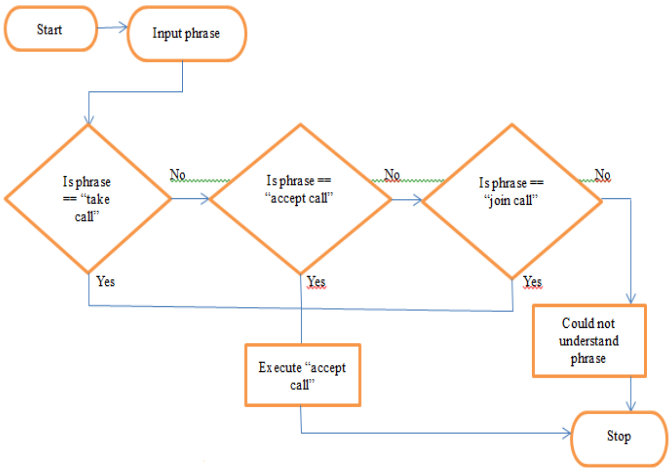

Fig -1: Illustration of rule based approach

Proposed system is creating bots that use Machine Learning-based approach that make them more efficient than rule-based bots. These bots are also known as Retrieval based models**.** These bots are trained on a set of questions and their possible outcomes. For every question, the bot can find the most relevant answers from the sets of all possible answers and then outputs the answer. Although, the bot cannot generate new answers if trained on a lot of question and answer dataset, and if the data set is pre-processed smartly, the bot can handle queries fairly good. The complexity can range from simple rules for a query to complex rules using some machine learning algorithm to find the most appropriate answer. Also, there is no issue with the language and grammar as the answers are pre-determined and it cannot go wrong in syntax manner. Our system uses machine learning model called memory networks [5]. Memory network uses inference components combined with a long term memory component and jointly it is used for predicting the responses. Memory network is trained end to end[4] and hence requires less supervision during training, thus it can be used for realistic settings. All the components of the end-to-end systems are trained on past dialogs, so it is not domain specific, thus it can be automatically scale to new domains.

2. RELATED WORK

This section starts with an introduction of the primary elements of MemN2N used for Question answering. Then, we review another area relevant to this work, namely

memory dynamics in such models. Also the implementation of goal oriented dialog systems.

2.1 End-to-End Memory Networks

The MemN2N architecture, introduced by Sukhbaatar[4], consists of two main components: supporting memories and final answer prediction. Supporting memories are in turn comprised of a set of input and output memory representations with memory cells. The input and output memory cells, denoted by m_i and c_i , are obtained by transforming the input context $x_1 , \ldots , x_n$ (or stories) using two embedding matrices A and C (both of size d × |V| where d is the embedding size and |V| the vocabulary size) such that $m_i = A\Phi(x_i)$ and $c_i = C\Phi(x_i)$ where $\Phi(\cdot)$ is a function that maps the input into a bag of dimension |V|. Similarly, the question q is encoded using another embedding matrix B $\in R^{d\times|V|}$, resulting in a question embedding $u = B\Phi(q)$. The input memories $\{m_i\}$, together with the embedding of the question u, are utilized to determine the relevance of each of the stories in the context, yielding a vector of attention weights

$$p_i = \text{softmax}(u^T m_i) \qquad (1)$$

where $\text{softmax}(a_i) = \frac{e^{a_i}}{\sum_{j\in[1,n]} e^{a_j}}$. Subsequently, the response o from the output memory is calculated as:

$$o = \sum_i p_i c_i \qquad (2)$$

For more difficult tasks requiring multiple supporting memories, the model can be extended to include more than one set of input/output memories by stacking a number of memory layers. In this setting, each memory layer is named a hop and the $(k + 1)^{th}$ hop takes as input the output of the k^{th} hop:

$$u^{k+1} = o^k + u^k \qquad (3)$$

Lastly, the final step, the prediction of the answer to the question q, is performed by

$$\hat{a} = \text{softmax}(W(o^k + u^k)) \qquad (4)$$

where â is the predicted answer distribution, $W \in \mathbb{R}^{|V|\times d}$ is a parameter matrix for the model to learn and K the total number of hops.

2.2 Memory Dynamics

The necessity of dynamically regulating the interaction between the so-called controller and the memory blocks of a Memory Network model has been study in [2]. In these works, the number of exchanges between the controller stack and the memory module of the network is either monitored in a hard supervised manner in the former or fixed apriori in the latter. In this paper, we propose an end-to-end supervised model, with an automatically learned gating mechanism, to perform dynamic regulation of memory interaction.

2.3 Goal Driven Dialog System

Goal driven dialog system is a system based on a specific goal. Perhaps the most successful approach to goal-driven systems has been to view the dialogue problem as a partially observable Markov decision process (POMDP)[14]. Most deployed dialogue systems use hand-crafted features for the state and action space representations, and require either a large annotated task-specific corpus or a horde of human subjects willing to interact with the unfinished system, hence limits its usage to a narrow domain. Our aim is to generate random conversations for goal oriented scenarios. This is equivalent of user simulators that are used to train POMDP [14].

3. PROPOSED SYSTEM

Our proposed system is a hospital appointment system, where the goal is to get the doctor appointment. Model is trained with training data that contain input user utterances with expected output. In an interactive environment user can chat with the bot in his own language, since we are using memory network rather than rule based, dialogues are context specific and bot give appropriate responses.

Proposed system uses gated end-to-end memory networks model. This is an end-to-end supervised model with an automatically learned gating mechanism to perform dynamic regulation of memory interaction [11]. This will replace hard attention mechanism in Memory networks where attention values are obtained by softmax function. Gated End-to-End Memory network uses the idea of adaptive gating mechanism of Highway Networks and integrate it into MemN2N. Highway Networks, first Highway Networks, first introduced by Srivastava [3], include a transform gate T and a carry gate C, allowing the network to learn how much information it should transform or carry to form the input to the next layer. The generic form of Highway Networks is formulated as:

$$y = H(x) \odot T(x) + x \odot C(x) \qquad (5)$$

where the transform gate T(x) and carry gate C(x), are defined as non-linear transformation functions of the input x and ⊙ the Hadamard product. Here we are using simplified form of Highway network where carry gate is replaced by 1-T(x):

$$y = H(x) \odot T(x) + x \odot (1-T(x)) \qquad (6)$$

where $T(x) = \sigma(W_T x + b_T)$ and σ is the sigmoid function

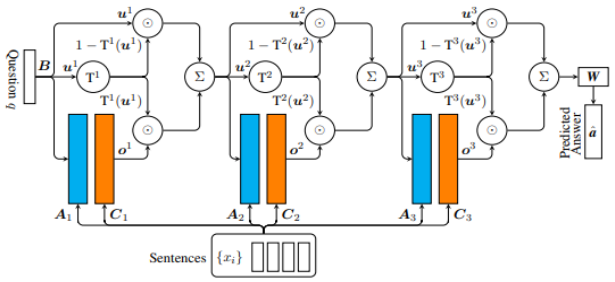

Fig -1: Illustration of Gated End-to-End memory networks

3.1 Memory Representation

As our model[1] conducts a conversation with the user, at each time step t, memory is formed by appending previous utterance from the user($c_1^u, \dots, c_t^u$) and responses from the bot($c_1^r, \dots c_{t-1}^r$), ie the entire conversation . At time t, model needs to choose the next bot's response c_t^r. We train on about 5000 full dialog transcripts in the training dataset, so at training time the model knows the upcoming utterance c_t^r and the target response. All conversations can be represented as a bag-of-words and in memory it is represented as a vector using embedding matrix A. So the memory can be represented as:

$$m=(A\Phi(c_1^u),A\Phi(c_1^r)\dots,A\Phi(c_{t-1}^u),A\Phi(c_{t-1}^r)) \quad (7)$$

where $\Phi(\cdot)$ maps the utterance to a bag of dimension V (the vocabulary), and A is a d × V matrix, where d is the embedding dimension. Last user utterance cut as the "input" that is used directly in the controller.

3.2 Attention over the memory

The last user utterance c_t^u is embedded using the same matrix A giving q = $A\Phi(c_t^u)$, which can also be seen as the initial state of the controller(u^k). At this point the controller reads from the memory to find intended parts of the previous conversation that are relevant to produce a response. The match between q and the memories is computed by taking the inner product followed by a softmax: $p_i = softmax(u^T m_i)$, giving a probability P vector over the memories. The vector that is returned back to the controller is then computed by $o=R\sum_i p_i m_i$ where R is a d × d square matrix. The gated end-to-end memory network is capable of dynamically conditioning the reading mechanism of memory controller state u^{k+1} at each hop. The controller is updated as below for a fixed number of iterations N(termed N hops).

$$T^k(u^k)=\sigma(W_T^k u^k + b_T^k) \quad (8)$$

$$u^{k+1} = o^k \odot T^k(u^k) + u^k \odot (1-T^k(u^k)) \quad (9)$$

where W_T^k and b_T^k are the hop-specific parameter matrix and bias term for the k th hop and $T^k(x)$ the transform gate for the k th hop.

3.3 Choosing the response

The final prediction is then defined as:

$$\hat{a}=\text{Softmax}(q_{N+1}{}^T W\Phi(y_1),\dots,q_{N+1}{}^T W\Phi(y_C)) \quad (10)$$

where there are C candidate responses in y, and W is of dimension d × V . In our tasks the set y is a candidate file that contains set of bot responses. The entire model is trained using stochastic gradient descent (SGD), minimizing a standard cross-entropy loss between â and the true label a.

4. EXPERIMENTS

In this section, we first describe the natural language reasoning dataset we use in our experiments. Then, the experimental setup is detailed. Lastly, we present the results and analyses.

4.1 Restaurant Reservation Simulation

Hospital appointment system is based on a set of facts and these facts may contain a specialization (5 choices eg., Dentist, Physician), doctor name under each specialization(5 choices eg., Roy, Madhu), appointment day(2 choices eg., today, tomorrow), patient profile. Since we cannot include all details of patient in dataset, we may add gender and age details (gender: 2 choices eg., male, female age: 5 choices eg., infant, child, young, middle-aged and old). Each doctor is assigned to a list of token numbers from 1 to 20 listed in the knowledge base. API calls are generated that consists of five fields eg., specialization: Dentist, doctor name: Dr Teena, appointment day: today and patent profile: male young. These API calls are saved in hospital database for each doctor appointment and a token number is also generated for each appointment. Conversations are generated randomly using these KB as in the format shown in the below figure. Each sample is a dialogue comprising utterances from a user and a bot, API calls and token number and these are all natural language pattern. There are 43 patterns for the user and 20 for the bot. Patterns means user can use 4 different ways to express something eg., can you make an appointment, I want an appointment, I would like to book a doctor, can you book a doctor, but bot uses same response eg., I'm on it, sure is there anything else to update. Those patterns combined with KB entities to form five thousand different samples of training data, validation data and test data.

4.2 Datasets

Datasets for training, validation and testing are generated by randomly running scripts. These three are mutually exclusive datasets. Validation dataset is used to minimize overfitting as training dataset actually produces an increase in accuracy. A user request implicitly forms a query that contains the required fields for API call. The bot must ask questions for filling the missing fields and eventually generate the correct corresponding API call. The

bot asks questions in a deterministic order. Users then ask to update their requests between 1 and 3 times except for specialization. The order in which fields updated is random. The bot can confirm with users whether they are done with the updates and issue the updated API call. Token number is generated for each appointment. Users always accept the token number and conversation ends.

Knowledge Base(KB) consists of 4,200 facts and 500 appointments each (5 specializations × 5 doctors × 2 days × 2 gender × 5 age groups) and the whole dataset contain disjoint sets of conversations. Training dataset contain 5,000 samples to create realistic learning conditions. All bot responses are stored in another file called candidate file, this file contain the labels. During testing we check if models generate new combinations of fields. Dialog systems are evaluated in a ranking and at each turn of the dialog candidates from candidate file are ranked from set of all bot utterances and API calls shown in training, validation and test sets. Thus it predicts bot utterance and API calls. Even though the model is trained with patterns for eg., can you make an appointment, I want an appointment, I would like to book a doctor, can you book a doctor, if user uses different pattern like please make an appointment or I need an appointment, the bot understand the context behind the dialogue and provide appropriate response. This is achieved by using model with multiple hops over the long term memory.

As in [4], our model is fully end-to-end trained without any additional supervision other than the answers themselves.

4.3 Data Pre-Processing

Next step is the tokenization of dialogs from the datasets. For performing tokenization, user and bot responses from the above files are separated and store into two lists u and r and perform tokenization. Then create another list context and append tokenized user utterance and bot response separately. This will constitute a conversation. Then create another list data and store previous conversation from context and current user question and append index from candidate file for the previous bot response. The tokenization process should do for all three files. So the final data list constitutes tokenized user utterances and bot responses with proper index from candidate file. Then a vocabulary or dictionary is created using the wordings used in the conversation and index also added to the vocabs. Then vectorize bot responses from candidate file based on the index in vocabulary.

4.4 Training

Learning rate η is initially assigned a value of 0.001 with 500 epochs. Linear start is used in all our experiments as proposed by [4]. With linear start, the softmax in each memory layer is removed and re-inserted after 10 epochs. Batch size is set to 32 and gradients with an 2 norm larger than 40 are divided by a scalar to have norm 40. All weights are initialized randomly from a Gaussian distribution with zero mean and σ = 0.1 except for the transform gate bias b_T^k which we empirically set the mean to 0.5. Only the most recent 50 sentences are fed into the model as the memory and the number of memory hops is 3. In all our experiments, we use the embedding size d = 20. Optimizer used is Adam Optimizer. We also constrain ourselves to the hop-specific weight tying scheme in all our experiments since GmemN2N benefits more from it than global weight tying.

4.5 Results

While per-response accuracy calculates the percentage of correct responses, per-dialog accuracy, where a dialog is considered to be correct if and only if every response within it is correct, counts the percentage of correct dialogs.

Accuracy is calculated by following equation:

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

Where $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value over n_{samples}. We passed over actual answer and predicted answer over sklearn. metrics function and received below accuracy.

Training Accuracy: 0.866678031242

Validation Accuracy: 0.856304621164

Below screen shots shows examples of predictions of our model for test examples.

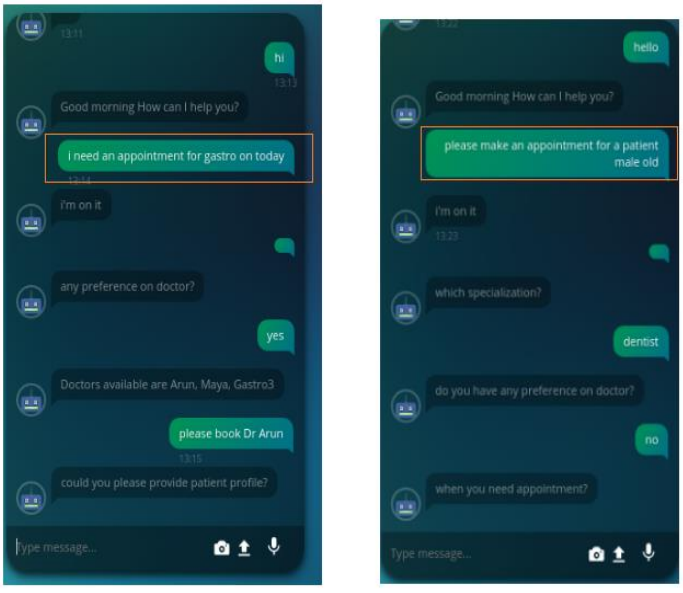

Fig -3: Shows that the model identifies the language context and then reacts based on that intend, whereas rule based approach understands a pre-defined set of options.

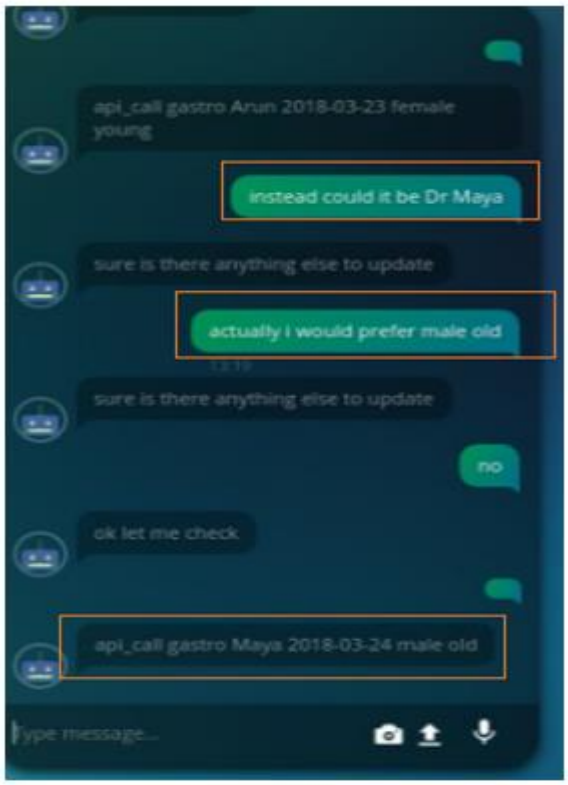

Fig -4: Shows that api calls are properly updated based on the user preferences

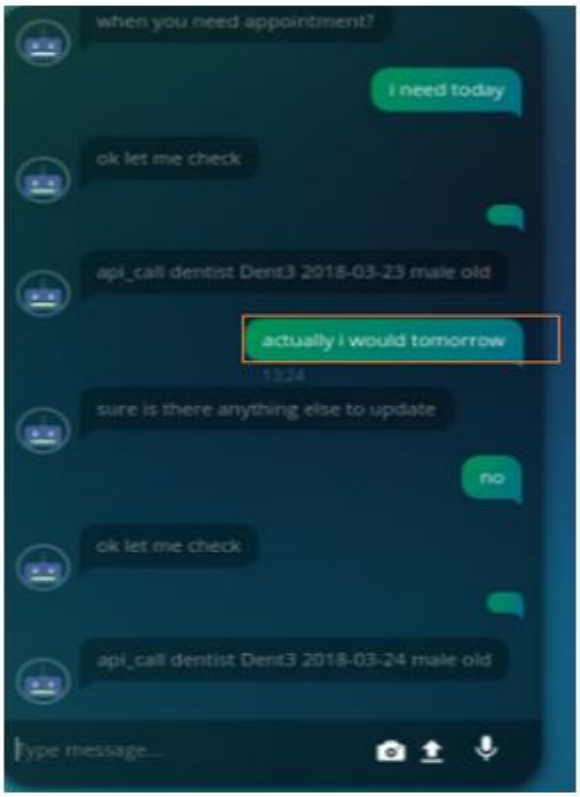

Fig -4: Shows that the model is able to ignore grammatical errors

5. CONCLUSION AND FUTURE WORK

In this paper we have proposed an attempt of incorporating an iterative memory access control to an end-to-end trainable memory enhanced neural network architecture. We showed the advantage of dynamic regulation of memory interaction, but it is still lack in some areas. In future work, we will investigate our model in language modeling and predictive learning so that the machine is more intelligent in goal based applications even though we are using synonyms or out of vocabulary words during conversation.

REFERENCES

[1] A. Bordes and J. Weston. Learning end-to-end goal-oriented dialog. In Proceedings of ICLR.March 2017

[2] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, and R. Socher, Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In Proceedings of the 33rd International Conference on Machine Learning (ICML 2016), New York, USA

[3] R. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. In ICML 2015 Deep Learning workshop

[4] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. 2015. End-to-end memory networks. In Proceedings of Advances in Neural Information Processing Systems (NIPS 2015), pages 2440–2448, Montreal, Canada.

[5] J. Weston, S. Chopra, and A. Bordes. 2015. Memory networks. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, USA.

[6] J. Weston, A. Bordes, S. Chopra, A. Rush, B. Merrienboer, A. Joulin and T. Mikolov. 2016. Towards AI-complete question answering: A set of prerequisite toy tasks. In Proceedings of the 4th International Conference on Learning Representations (ICLR 2016), San Juan, Puerto Rico.

[7] C. Xiong, S. Merity, and R. Socher. 2016. Dynamic memory networks for visual and textual question answering. In Proceedings of the 33rd International Conference on Machine Learning (ICML 2016), pages 2397–2406, New York, USA.

[8] R. Srivastava, K. Greff, and J. Schmidhuber. 2015b. Training very deep networks. In Proceedings of Advances in Neural Information Processing Systems (NIPS 2015), pages 2377–2385, Montreal, Canada

[9] A. Miller, A. Fisch, J. Dodge, A. Karimi, A. Bordes, and J. Weston. 2016. Key-value memory networks for directly reading documents. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016), Austin, USA.

[10] Y. Chen, D. Hakkani-Tür, G. Tur, J. Gao, and L. Deng, (2016). End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In Proceedings of Interspeech.

[11] F. Liu, J. Perez (2017). Gated End-to-End Memory Networks. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics(ACL 2017), Spain

[12] H. Wang, Z. Lu, H. Li, and E. Chen, (2013). A dataset for research on short-text conversations. In EMNLP.

[13] Z. Wang, and O. Lemon, (2013). A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In Proceedings of the SIGDIAL 2013 Conference.

[14] S. Young, M. Gasic, B. Thomson, and J. D. Williams, (2013). Pomdp-based statistical spoken dialog systems: A review. Proceedings of the IEEE,101(5), 1160–1179.