SmartGator: Multi-domain Chatbot

Akshay S. Jetawat
(akshayt80@ufl.edu)
Dept. of Computer Science & Engg.

Karan Goel
(karangoel16@ufl.edu)
Dept. of Computer Science & Engg.

Vishal K. Gupta
(vishal191@ufl.edu)
Dept. of Computer Science & Engg.

***Abstract*—Deep Learning has proven to be very efficient in solving a range of issues faced by previously used artificial neural networks, one application being language translation deployed in chatbots. Now a days chatbots are bringing new era of putting all the apps in one place. Applications of chatbots are boundless. We are going to use LSTM network, a variant of RNN, at the backend of our chatbot and do the conversation with users on any platform. We will train it over 2 different data sets: Cornell Movie Dataset and OpenSubtitles Dataset.**

***Index Terms*—RNN, LSTMs, Chatbot, Sequence-to-sequence, Learning, ANN, Deep Learning, Attention Mechanism**

I. Introduction And Motivation

For the past decade, mobile industry has been growing at a tremendous speed. It's also undeniable to overlook the multitude of problems undertaken by current advancements in AI. It's obvious to see that the next revolution is going to be something where mobile devices are will be self-reliant in their assistance to user on account of huge amount of data and frameworks available for their learning. Having said that, chatbots looks like one obvious step towards this phase of user-smartphone interaction, where a user can fluently type/say his/her views and requirements, instead of selecting from one of the many pre-coded options, and device take steps accordingly. A vast majority of applications already exist where chatbots are assisting users on day to day basis: shopping bots, bots which can assist patients with disorders and problems of anxiety, messenger chatbots like the one used in Hike, google assistant etc. Since the constraints exist for memory in phones, chatbot could overwhelm apps like twitter. An important observation regarding the existing chatbots is their specificity: they entertain a certain domain or become evasive in their answers by replying a very generic way: *"Can you provide more information"* and things like that! We plan to incorporate more than one domain knowledge. As a proof of concept, we wish to deploy our model at the backend of a tweetbot also in the scenario where the texts are constrained within a limit of 140 characters and conversation happening over multitude of domain.

Past decade has seen resurgence in the application of Deep Learning for a host of tasks. This is attributed to the fact that deep architectures are able to capture hidden patterns which shallow architectures overlook. Despite the powerful approach of feedforward DNN and CNN, we still can't use it for language translation and modelling because of three very crucial characteristics of conversation: *i) state connectedness* (structure of sentence); *ii) context* (relation with previous replies/statement); and *iii) variable length input* (# of words in a sentence). Recurrent Neural Networks (RNNs), which effectively loopback current state, are able to solve 1st and 3rd issues. For solving the 2nd issue, various tricks are used. One or two will also be mentioned here. We aim to use LSTM based RNN architecture with the end goal of training a chatbot which can converse in more than one knowledge domain. We have found many related works, however, to the best of our knowledge, most of them are domain specific.

The rest of the project report is structured as follow: following *related work*, first we list the basics of RNNs establishing the fact that how it is effective in language modelling. Then we provide overview of our approach in context of targeted tasks. Next section discusses the diversity of our dataset we are going to use, which will ensure our chatbots are intelligent in more than one knowledge domain. Then follows some general discussions and potential challenges we faced while achieving our end goal.

II. Related Work

In context of chatbots, we have found a host of applications and independent projects by individuals on github and various other online sources deploying one or other versions of RNN at the backend of their chatbots [16][17][18]. The more interesting part of literature review was the backend learning models of various chatbots, which basically comprises of sequence learner/predictor, typically an RNN model or its variant. Other aspect of conversation generation and learning is context determination (for eg sentiment analysis). Below we list down some recent literature reviews pertaining to our project.

The use of LSTM RNN for temporal sequence modelling is found to be far more effective as compared to simple RNN or conventional DNN [10][19]. Moreover, the task of machine translation is accomplished in a better fashion using an encoder-decoder RNN network[2][3]. The prime idea

Code:https://github.com/karangoel16/BIG_DATA

being: train the encoder and decoder network via maximizing the conditional probabilities of phrase pairs i.e. target sequence & source sequence over existing log-linear model. However, the issue of vanishing and exploding gradients along with learning time complexity is quite vibrant with LSTM RNN. But Hinton et. al., in their work[20], show that using ReLU along with careful initialization of recurrent weight matrix instead of sigmoidal/tanh activation omits this drawback. At later stage of our project, we wish to exploit ReLU in our LSTM cells.

In our research we came across very advanced bots of amazon, google, microsoft, that bolster our belief that the entire industry is shifting towards the chatbots and it is very exciting to see that even. Reference section list down many relevant works (not explained here) [16][17][18].

III. Deep Learning Architectures for Sequential and Variable Length Inputs

Language learning doesn't happens from scratch, it proceeds via persistence. As far as traditional ANN is concerned, they don't have any kind of memory/loopback to account for this. For this reason, RNN came into existence.

The following fig.1 [9] shows a single RNN cell and when its unfolded:

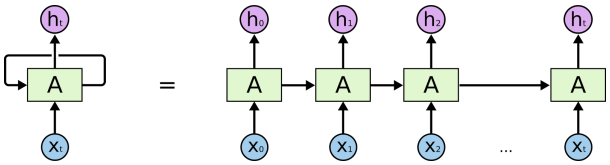

fig.1 single cell of RNN and its unfolded version

As obvious from the diagram, the loopback *loops into* some information about the current h_t back into A for computation of h_{t+1} from x_{t+1}. The persistence relation can be summarised as follow: $h_{t+1} = A(x_{t+1}, h_t)$. In practice, when the relevant (past) information in too old, RNN fails to connect the information. In theory it is possible, but [11] explains why it fails.

A. Unit Block of Network

Long Short Term Memory (LSTM) network [10], a variant of RNN efficiently solve the problem of *long-term dependency*. They exhibit the same chain line structure but repeating cell are a bit complex consisting of four interacting layers. It has three sigmoidal gates, which output numbers between zero (let nothing pass) and one (let everything pass); decide how relevant is the past state for generating the current state. The *tanh* layer discovers new candidate values that would be added to current state. The final output consists of two things: sigmoidal output of x_t (parts of cell state we are going to output) and push it through *tanh* which essentially outputs the part we decided to [9].

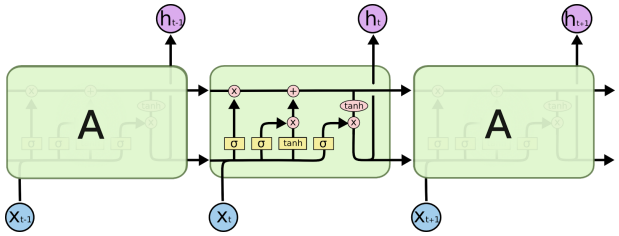

Fig.2 LSTM cell architecture

There are many variants of LSTMs, for eg. peephole connections [12], coupled forget and input gates and GRU [2]. All the variants capture one aspect or another, we will look out which of the variations, one or more, works best for our task.

B. Dataset and PreProcessing

In this section we list down our pre-processing steps and details of datasets we are going to use. We are preprocessing the entire corpus and putting it in a standard format. Precisely, the format is as follow:

Preprocessing also include tokenization of the data and then building & maintaining the dictionary of corpus using NLTK. Since, the format of datasets is different for all different datasets, we have created separate python scripts for each dataset to make them consistent with our input format (explained above), As a check, we tried to retrieve words from their IDs and vice-versa and found it ok, For now we have validated two datasets: cornell and scotus corpus, and are working on ubuntu. Size of datasets present yet another challenge, we are not able to work with it on our computer as Ubuntu dataset only is around 1GB, and thus require the computational power, which is not possible on our computers and we plan to do so in the second phase of the project , though the stuff and initial backbone related with these three corpus is ready, unit testing is successful, and process integration will eventually validate the correctness of our data format.

During the course of our experiments, we found that, it is hard to maintain context in the above format as there is no relation, whatsoever, among consecutive inputs. The results that we obtained (explained in results section and discussion) also reflect this. This is also surprising to know that, not much effort has been made in this regard among research communities also. So, we did a small trick. This doesn't guarantees long term context linking but short term is intuitive. The trick is as follow: instead of taking two statements (perceive as a question-answer pair) at a time, we

Code:https://github.com/karangoel16/BIG_DATA

considered three consecutive statements at a time, combined the first two as one (*new input*) and the third one as expected output of the new input. It's quite obvious to see, how contexts are linked in short term.

C. Network Configuration

Our baseline model comprises of two RNN layers with 256 LSTM blocks as their building uint. We are using Adam optimizer from TensorFlow API and one softmax at the end to output words having maximum posterior probability conditioned on input. In later stages, we experimented with three layers and 512 LSTM cells in one layer. Due to hardware limitation and time it took to train a single model, we were unable to go beyond this configuration. On the hyperparameter side, we have kept our learning rate to 0.001 as initial experiments suggested this as the best. We have tried with different batch sizes also. In all batch size of 32 and 10 gave us the best results.

In case of attention mechanism, the parameter to vary is the embedding size. In our baseline model with attention mechanism, embedding size is 256. Later models had embedding size of 512 with 512 LSTM cells in one layer and 3 layers. Again due to hardware and time limitation, we were unable to go beyond this configuration.

D. Integrated Framework

We will expose a single interface with backend framework, denoted as *interaction interface*, through which our model will receive input in the form of a semantically correct meaningful statement or output predicted reply from chatbot. Our backend framework comprises of two modules: *dataset module* and *learner/prediction module*. The input from interaction interface will go into dataset module where it would be encoded into a sequence of IDs as per the dictionary. Then these pattern of numbers would be sent to the the learner/prediction module. The network perceives this sequence as a statement, learns/extracts the necessary semantics and predicts a sequence of IDs. This new sequence is then forwarded to dataset module to decode it as another statement for output to interaction interface.

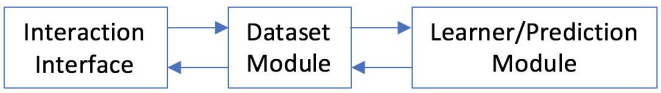

Fig.3 Flow Diagram

E. Attention Mechanism

Attention mechanism is motivated from visual attention mechanism of human beings: *we humans can focus on a relatively small portion of an image in high resolution and surrounding regions in low resolution*. This concept can be deployed in a straightforward manner in case of images. The application of this in NLP is quite recent [25]. Language is a

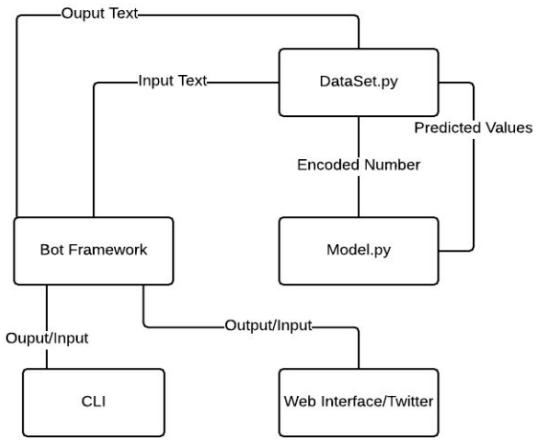

Fig.4 Class Diagram Representation

complex structure where any word could have more value than other and sequence of words while replying to a query also matters. Literature shows that, even LSTMs are not sufficient to capture very long dependencies. The trick of reversing the sequence is quite new: works but it's not a principled solution. In short, it's a hack! However, attention mechanism promises better results. In essence, it does not encode an input sequence into a fixed length vector but allows the decoder to *"attend to"* various parts of input sequence at each step. For eg, if initial words of a reply to a query is supposed to be linked with initial words of query, decoder will refer to those words in order to produce the most apt words. During training, the model learns these patterns: how to attend to different parts of of input. This is equivalent to the visual attention mechanism of humans, our eyes automatically adjusts to that.

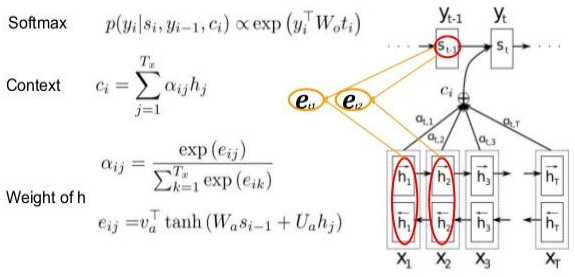

Fig.5 Representational Attention Mechanism

However, attention is costly, how? The models needs to figure out the attention cost of each combination of input and output. For eg. if input sequence is 10 words long and output is also 10 words long, the model has to calculate 100 attention costs. That's why when we moved from tokenizer model to attention mechanism model, our model exploded. However, we still able to train our model with a small trick (refer to compute management section). The perplexity plot

Code:https://github.com/karangoel16/BIG_DATA

in results section clearly depicts how our model improved with attention mechanism over seq2seq model. In later sections, we have argued why we were not able to converge when we used the se2seq model where we fed in the 3 way conversational model, but attention mechanism worked well.

IV. Application and Usability

The outcome of this project achieves two fold usability: 1) as a backend framework for a host of applications; 2) the learning paradigm of backend framework itself can be applied to solve problems in other domain.

Once, we have a effective chatbot backend framework, we can train a single bot for a host of applications instead of having multiple agents for different applications. If the network is able to accommodate more than one domain knowledge, it can be made more deft in its prediction with online learning paradigm easily as the problem of extracting relevant domain data vanishes.

The multi-domain feature, which we are trying to accomplish, however being a tough task, but the training methodology of the network will have some very fine implications. For eg. we can build a single integrated framework which can be used for language translation in more than one languages. A more generalized use cases include, context dependent predictions of temporal sequences from a list of possible predictions.

V. Data Management

We have been able to locate corpus of ubuntu [22], scotus(Supreme Court of The United States of America) [23], opensubtitles [24] and cornell movie dialogue [13] datasets,for initial training and testing of our chat application. The conversation is saved in the form of a list and save to file using pickle package in python 3.5. This has to be done once and the saved file can be used again and again in subsequent runs This approach saves a lot of time, as the preprocessing of the data, is done once for a given requirement. We are also exploiting the "save model" feature of TensorFlow API to save the model. This makes porting the model easy from one machine to another or use it separately during testing, independent of model training. Increasing the batch size does improve the result and reduced training time but it's not always beneficial. We tried to change batch size from 10 to 16 to 32 and then 256. Results improved from 10 to 16. The performance from batch size of 16 and 32 were comparable, but when we moved to 256, it deteriorated: results which we got were not really up to the mark. So, we concluded to stick with batch size of 16 in further experiments.

Variation in vocabulary size also affects the quality of replies that the bot outputs. Too less vocabulary size limits the diversity in context and too high vocabulary size increases the training time of bot by a very large factor in addition to introducing noise because it is highly unlikely to have a dataset which can be considered free of noise. We varied the vocab size from 5,000 to 30,000 and concluded that vocab size between 7,000 and 10,000 gives us satisfactory result with acceptable training time.

Once all the potential variations in the model has been experimented and concluded on the dataset on the following datasets: scotus and cornell; we moved to opensubtitles dataset. Earlier we planned to do on reddit dataset, but it is too much in volume to process, given the time constraint and hardware limitations. The reason we chose opensubtitles is because it is sufficient in size and context as it comprised of many tv series/movies belonging to different genres. We also experimented with ubuntu dataset in mixture with scotus and cornell dataset but didn't concluded it. Reason being, the model started performing in a biased manner due to the volume of ubuntu dataset. because of same two reasons mentioned above.

VI. Coordination Strategy

We have built our whole bot framework in a bottom up fashion using a modular approach. That is to say, each of the main modules, whether it is backend i.e. dataset management, model design, model training & testing; or frontend i.e. twitter interface have sub-modules to micro-manage their function. This provided us with many advantages:

- Any of the modules can call to any other functionality of other modules as required
- Validating the approach at the finest level can be achieved
- Debugging was cakewalk

Furthermore, each module can be individually tested by giving appropriate inputs and calling on a main function which validates all other sub-modules in that module. In short, our code is so well written and organized that anyone can pick it up and build their applications on top of that. As a proof of concept, our twitter interface displays the fine abstractions elegantly. Whole of twitter interface is independent from backend. We make a single call to a sub-module defined in our training & testing module which handles both way communication i.e. feeding input sequence to backend and retrieving the reply. So not only for this application, our twitter module can be picked up and put at

Code:https://github.com/karangoel16/BIG_DATA

the front end of any other application with minimal amount of coding required to bring up the frontend-backend communication

VII. Compute Management

It's not a hidden fact that all deep architectures are compute intensive. When it comes to language models, the situation worsens. To provide a sneak peek, consider the case of word2vec model. If we need sufficient representation of a single word with a decent vocabulary size (say 7-10k), the word needs to be 300-400D. The dataset is tremendous in size. If the vocabulary has more words, we need to have even larger dimensional representation of word. Model params adds further demands to the available hardware. So, efficient management of compute capacity is one of the important goals in any big data project, particularly ours.

Coming to our project, we had at our disposal around 10 Gigs (Tesla K80, thanks to Prof. Li.) GPU card (although it seems a lot) however, it was not sufficient as compared to the voluminous size of our model. We also had 100 Gigs of CPU memory. So, for efficient training, we pulled up a trick supported by TensorFlow API, known but used not very often. We initialized all our models (params/weights) on CPU memory and did all the gradient calculations on GPU. Obviously, this introduced additional overhead of copying data to and fro from CPU and GPU, but it furnished the requirement of high memory demand and we were able to train large models. The code provided is customizable in this respect also as deem fit to user. This trick proves more beneficial when we have multiple GPUs.

VIII. Performance Evaluation

We are going to train our model over three different data sets. One data set contains conversations from various movies [13]. We chose movie data set as it will be covering wide variety of conversations occurring in real life. Another data set we are going to use will be opensub conversations. Opensub because it contains movie/tv serial conversation for many genres and we believe that, this will provide us sufficient diverse domain knowledge. We will compare the performance of various models based on loss trajectory, replies to set of basic questions and human perception (demo during poster presentation).

We will use these data sets to train our model to conversate with people. We can apply our chat bot on any platform such as Twitter etc. to perform the conversation. We will have to do preprocessing on movie data set to extract information relevant to training.

In the initial planning phase we wished to train our model on reddit conversation but skipped it due to following reasons: dataset was too large to be handled by available hardware resources; and the training time model grew exponential as we have to scale up the model to incorporate diversity.

In the process of development we would be using NLTK [14], an open source Python library to perform text processing. "It provides easy-to-use interfaces to various corpora and lexical resources along with text processing libraries for classification, tokenization, stemming, tagging, parsing and semantic reasoning." as given here [14].

IX. Results

We implemented seq2seq model along with word2vec with the perception of achieving exceptional results. Instead what we discovered that seq2seq is good enough to give contextually correct sentences , but at times it does give elusive answer(i don't know). Since this area is very niche and this model is currently state of the art for the unsupervised learning. Literature survey revealed that seq2seq model is not enough to have continued conversation maintaining the context at the same time [26], we need to reinforcement learning. Next we experimented with attention mechanism which gave better results than normal seq2seq model, with respect to context of current sentence only. Still our model lacked context specific replies for long conversation. At the time of submission we have these models as best models. We didn't had enough time to experiment with reinforcement learning. So, we tried one trick of our own. We created dataset of individual samples consisting of 3 consecutive conversation from the dataset: 1 and 2 are merged together to form the input, and 3 is kept as expected output (reply) of input. This way we expect to predict not only semantically correct answers, but also we will be able to maintain the context over at least some past time stamps.

Below we show three plots for loss of our learned model for three models: tokenizer model (baseline model), word2vec model and attention mechanism model. Please note that the three shown plots are for similar training/testing environment which are as follow: batch size: 10; learning rate: 0.001; 256 LSTM cells; 2 layered network; trained over 30 epochs. A close observation of below plots reveal the following facts:

Code:https://github.com/karangoel16/BIG_DATA

- Word2vec model has more rapid drop in loss than tokenizer model, showing better convergence due to improved representation of dataset
- Word2vec plot is more smoother than tokenizer model, which establishes the fact that word2vec model is more stable and robust than tokenizer model.
- Although, hard to differentiate between exact difference in loss between the plots, word2vec las relatively less loss which further suggest the supremacy of word2vec model over tokenizer model.

Now we will do an equivalent comparison between word2vec model and attention mechanism model:

- The relative loss in the attention mechanism is less as compared to word2vec model which suggests the difference between expected output sequence and actual output sequence is less.
- The drop in loss is also consistent which suggest with increasing training steps, the model is able to predict better output sequences against input sequence.

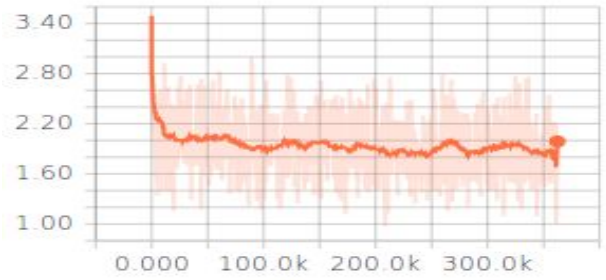

Fig. 6. Loss graph for cornell movie data corpus tokenizer model

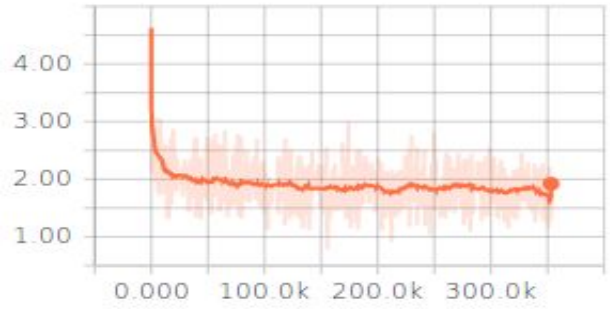

Fig. 7. Loss graph for cornell movie data corpus on word2vec model

We even figured out the bot is able to gain some characteristics of its own. For eg. we felt that the bot had submissive characteristics as when used “Idiot”, “Kill you” , the replies were mainly “I’m Sorry”, these characteristics of the bot doesn’t change a lot. We even found that the bot is confused at times and is not able to determine its gender. We list down this as limitation of current models, which is not specific to ours only (details in section X).

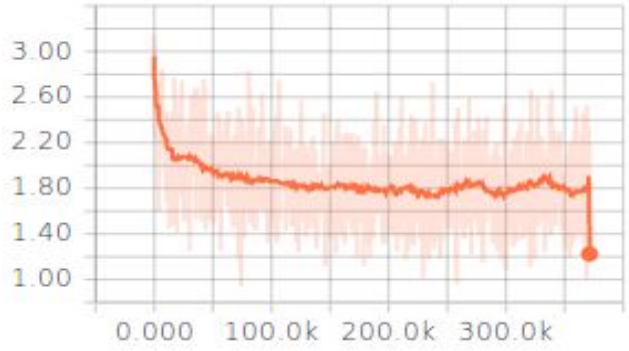

Fig. 8. Loss graph for cornell movie data corpus on attention mechanism model

After concluding that attention mechanism models performs best among all the three models, we scaled up our model to see if a deeper networks improves over current configuration. To our surprise, the model is able to improve drastically. Same is evident from the loss plot of scaled-up model:

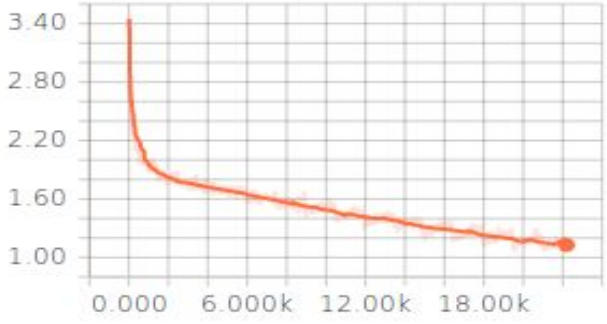

Fig. 8: Attention Mechanism loss plot with 2048 LSTM cells

As evident from the above plot, this scaled-up version of model is more stable(less zitter) and predicted sequences are more close to the expected ones (loss figures so obtained are lowest).

Now we list down replies of basic set of questions for different baseline models and compare the quality of replies. As we can see in table 1 and 2, simple tokenizer model failed to recognize the context of 4th question but word2vec model was successful. In total we had a set of 13 questions. Replies for all for each model is deliverable file. Comparing the replies, suggest, intuitively that reply from attention mechanism are most apt as expected. Other than these, there are other random queries put against the different models

Code:https://github.com/karangoel16/BIG_DATA

(present in deliverable file). All suggest results from attention mechanism models are best so far.

Question	Answer
Hi	Hi
Are you ill	I am sorry
Are you beautiful?	I am sorry
Good night	I'm sorry
Do you like red color?	Yeah

Table 1. Tokenizer model

Question	Answer
Hi	Hi
How are you?	I'm fine.
Are you beautiful?	I am sorry
Good night	Good night
Do you like red color?	I am not sure

Table 2. Word2Vec model

Question	Answer
Hi	Hi. How are you?
How are you?	I am fine
Are you beautiful?	I am not a good guy
Good night	Good Night
Do you like red color?	I like that

Table 3. Attention Mechanism model

We also observed overfitting in case of word2vec based model and attention mechanism model. The way we figured it out is the amount of time the code took while training. After many epochs (25 in case of word2vec model and 30-35 epochs in case of attention mechanism based model), we observed that, the amount of time to run 100 steps became constant i.e. the model was not updating anything significant. This was also observed in replies given by bot. When it became overfit, it started giving replies *"I'm sorry"* for any and every kind of question. One way to tackle over-fitting is dynamically change graph (by adding some more cells) and continue the training. One advantage of this trick is that, it will make our model deeper and sharper with time. This fact is just put intuitively, testing remains at large. Another trick which could help is add some more unseen sequences to the current dataset and thereby feeding more information to the model with time.

X. Discussion, Challenges & Future Work

The first and foremost challenge we faced is the non-availability of standard datasets. Each of the models out there picked up a conversation (say movie, court room etc.) and formulated their dataset in their own way. We also did the same (details in dataset section). We first trained our baseline model on movie conversation, though it has its own set of issues such as more number of participants rather than just two way conversation. Also sometimes the monologue when only one participant in conversation existed. Lastly but not the least, the conversation had no tagging that whether it is between persons of opposite gender or same gender. Hence this trained bot will output conversation from gender-neutral point of view. Other challenges pertaining to dataset is lack of context tagging. If each line in the conversation would have contained a tagged context (vague only would have worked), context learning can be boosted. These problems can be taken up as future work also. For instance, by means of manual tagging, we can remove gender ambiguity. As fas as context is concerned, one obvious approach is to design a hierarchical dataset where all the conversations could be grouped as per relevant scene (consider a movie) and then those scene could be grouped into precise contexts. Although quantity of contexts can be exponential but hierarchical classification would work there.

We preprocessed the ubuntu data and tried to train our bot with a mixture of the other datasets, our chatbot (supposed to learn multi-domain) got biased towards the ubuntu dataset and replies started inclining towards ubuntu dataset due to relatively high volume of ubuntu dataset. Hence, we refrained from further training on ubuntu dataset.

The model architecture can also be further improvised for continuous stream of data. The functionality of receiving data from and replying to a twitter handle is already there. The same can be modified to direct the incoming sequences and its replies to the training model and so that chatbot improvises in its responses in real time. The reason we skipped this part is because we focused on improving the backend of our chatbot, as it was a rigorous task explained in

Code:https://github.com/karangoel16/BIG_DATA

next paragraph. There is one more issue with twitter dataset: since there is character limitation, people try to optimize their content by using some non-standard but concise vocabulary. We need to have either an extended dataset or a model which maps these non-standard vocabulary to words understandable by model.

It is completely understandable that, the responses generated by our model, although syntactically correct and makes sense to some extent, but are not up to mark. However, it's not that that we lacked in our understanding and implementation. In essence, BUILDING A CHATBOT IS HARD! In order to have a chatbot of optimum quality, it should satisfy these basic premises: i) machine need to understand what we are asking; ii) machine need to generate coherent, meaningful sequences *in response to* what we ask; which means machine need to *identify & extract domain knowledge, discourse knowledge, world knowledge* from input and process them to output appropriate sequences [26]. The amount of bare minimum task required are in themselves contains unsolved problems. Before deploying attention mechanism, our bot was struck with *"I don't know"* problem, in case we asked it a tough question. Infact this is coherent with many other chatbot models out there. This can be explained: when the model has no context information of ongoing conversation, in essence it *"doesn't knows"* what to reply. Literature argues, a combination of seq2seq model and reinforcement learning can solve this problem: we need to tell model that expected output will get a reward to him and undesirable will penalize it with negative reward; that way model will be able to filter relevant domain knowledge [26]. In the short time span of this project, we were unable to experiment with reinforcement learning. However, in order to preserve context info, the three sentence model combined with attention mechanism did benefitted us. The compute intensive nature of deep architectures when combined with vast amount of data pose another problem. However, the situation is not that bad. Past decade has seen exponential increase in the number of problems being solved using deep architectures despite limitation. We can hope to have a chatbot in near future which can mimic human conversation to the best.

XI. Conclusion

This project served as a good learning experience for a wide range of topics starting from basic RNN to complex seq2seq models. This project can be put as baseline model and further improvements can be done on top of it to improve it. We would like to thank Prof. Andy Li for guidance and TAs for their continuous support in the completion of this project.

Code:https://github.com/karangoel16/BIG_DATA

References

[1] Kitchin, Rob, et al. "Category Archives: information."Jozefowicz, Rafal, et al. "Exploring the limits of language modeling." *arXiv preprint arXiv:1602.02410* (2016).

[2] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014).

[3] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." *Advances in neural information processing systems*. 2014.

[4] Jernite, Yacine, et al. "Simultaneous Learning of Trees and Representations for Extreme Classification, with Application to Language Modeling." *arXiv preprint arXiv:1610.04658* (2016).

[5] Kucukyilmaz, Tayfun, et al. "Chat mining: Predicting user and message attributes in computer-mediated communication." *Information Processing & Management* 44.4 (2008): 1448-1466.

[6] Abadi, Martín, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." *arXiv preprint arXiv:1603.04467* (2016).

[7] Shawar, Bayan Abu, and Eric Atwell. "Using dialogue corpora to train a chatbot." *Proceedings of the Corpus Linguistics 2003 conference*. 2003.

[8] Callejas-Rodríguez, Ángel, et al. "From Dialogue Corpora to Dialogue Systems: Generating a Chatbot with Teenager Personality for Preventing Cyber-Pedophilia." *International Conference on Text, Speech, and Dialogue*. Springer International Publishing, 2016.

[9] Github Tutorial: "http://colah.github.io/posts/2015-08-Understanding-LSTMs/"

[10] F. A. Gers and E. Schmidhuber, "LSTM recurrent networks learn simple context-free and context-sensitive languages," in *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1333-1340, Nov 2001.doi: 10.1109/72.963769

[11] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," in *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157-166, Mar 1994.doi: 10.1109/72.279181

[12] Gers, Felix A., and Jürgen Schmidhuber. "Recurrent nets that time and count." *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*. Vol. 3. IEEE, 2000

[13] Cornell Movie--Dialogue Corpus: "https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html"

[14] Edward Loper and Steven Bird. 2002. NLTK: the Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics - Volume 1* (ETMTNLP '02), Vol. 1. Association for Computational Linguistics, Stroudsburg, PA, USA, 63-70. DOI=http://dx.doi.org/10.3115/1118108.1118117

[15] Fan, Kuo-Chin, Sung-Jung Hsiao, and Wen-Tsai Sung. "Developing a Web-based pattern recognition system for the pattern search of components database by a parallel

computing." *Parallel, Distributed and Network-Based Processing, 2003. Proceedings. Eleventh Euromicro Conference on*. IEEE, 2003

[16] TensorFlow Chatbot: https://github.com/llSourcell/tensorflow_chatbot

[17] DeepQA: https://github.com/Conchylicultor/DeepQA

[18] Retrieval based conversational model in TensorFlow: https://github.com/dennybritz/chatbot-retrieval/

[19] Sak, Hasim, Andrew W. Senior, and Françoise Beaufays. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling." *Interspeech*. 2014.

[20] Le, Quoc V., Navdeep Jaitly, and Geoffrey E. Hinton. "A simple way to initialize recurrent networks of rectified linear units." *arXiv preprint arXiv:1504.00941* (2015).

[21] Garcia, M. (2016). Racist in the Machine The Disturbing Implications of Algorithmic Bias. *World Policy Journal*, *33*(4), 111-117.

[22] Lowe, R., Pow, N., Serban, I. and Pineau, J., 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

[23] Harold J. Spaeth, Lee Epstein, Andrew D. Martin, Jeffrey A. Segal, Theodore J. Ruger, and Sara C. Benesh. 2016 Supreme Court Database, Version 2016 Release 01. URL: http://Supremecourtdatabase.org

[24] Tiedemann, Jörg. "News from OPUS-A collection of multilingual parallel corpora with tools and interfaces." *Recent advances in natural language processing*. Vol. 5. 2009.

[25] Hermann, K.M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M. and Blunsom, P., 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems* (pp. 1693-1701).

[26] Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J. and Jurafsky, D., 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Code:https://github.com/karangoel16/BIG_DATA