TwinkerBot

Tapia de la Rosa Jacobo Misael
Instituto Tecnológico y de Estudios Superiores de Monterrey
Ciudad de México, México
jatapiaro@gmail.com

Abstract

With the advancements in artificial intelligence, and millions of human beings interacting online using social media applications like twitter or facebook; sooner or later the human-robot interaction will take place on this platforms. In this paper, we`re going to explore those interactions on Twitter. To achieve our purpose, I built a bot that emulates a normal user in twitter, hiding his artificial intelligence he will take the role of a human being, searching for other users who share his interest, write tweets about what he likes and interacting with other user tweets.

Author keywords:

Social Bot; Twitter; Natural language analysis; Turing test; Human-Bot interactions; Twitter Bot.

State of art:

The term "chatbot" was originally invented as a game character named **CHATTERBOT** for a multiplayer dungeon game, it has the objective to answer the players questions about navigation, other player and objects available in the world. In fact, it was only replicating a program called **Elisa**, created in 1966 by Joseph Weizenbaum, and his only purpose was to simulate a psychotherapist; users can interact with the program using natural language via typed. These systems used a stimulus response pattern-matching algorithm for the dialog functionality, determining the reaction using a database with a set of pattern-template pairs, if the input matched with an existing pattern, the system return an answer to the user.

Since then, many chatbots had been created, and many of them include a personality feature, that emulates a person of a certain age, sex and with a set of likes.

Two of those bots were made by Microsoft, but unfortunately one of them was taken out of the internet.

The first one is **Rinna**, which started as a bot for the instant message application Line in Japan, simulating a college student, who also use blogs and twitter to make comments about his life. But someday Rinna started to write depressing messages; but no one knows if it was just a publicity campaign or the bot really started to learn about how a schoolgirl in Japan feels and share his thoughts on the internet, so the system replicates it. It's interesting if the bot really replicates an emotional feeling based of what is learning; even if it scared the users of the platforms who interacted with de AI.

The second one, but this created specifically for Twitter, and again developed by Microsoft in the United States, was called **Tay**, because it was banned from the internet, as soon as the experiment to know better the interaction between human and computers fail when the bot started to making racist and xenophobic comments. Originally his objective was to have a funny and informal conversation with users between 18 and 24 years old. One day after his launch, was taken out of twitter for the comments; resurrecting a few weeks later just to be out again, this time because the system tweet so often (in question of seconds), again and again a set of predefined texts. Although she was funny at the beginning, the bots has no idea if hits tweets

are offensive, nonsensical or even profound; the bot started training with patterns input by trolls, and then it started to use them, like a parrot repeating words.

Another one and two times winner of the Loebner Prize for the world's most humanlike chatbot; **Mitsuku**, created by Steve Worswick, and it pretends to be a 18 year old girl from Leeds. Actually he won Siri in a chatbot smackdown. It's a really stable AI, the only way that she becomes mean is if you're mean with her, she learns from the behaviour and language of the people who talk with her. Another interesting thing is that she can share with you images, websites, and lot of things; and of course she can remember you if you're so nice to remember to say goodbye to her, so if you visited her again with the same computer, she will remember you. It's polite, even if you're mean she doesn't say horrible things, you can really have a nice chat with the AI although sometimes she will produce apparent meaning that could be considered inappropriate for some people.

Between the techniques used in this kind of agents, one of the most easy and common is the use of keywords which lead us to other keywords related to the first one to figure out an answer.

But in this context we have two different approaches, one is a **retrieval-base model**, that uses an heuristic to choose a response based on the input and the context, so they don't generate any new text, they just choose a response from a set of them. This approach is not able to handle cases where there's no an appropriate answer previously defined.

The other approach, which generate responses from scratch are the **generative models,** based on machine translation techniques, in which they "translate and input to an output". It's like a decision tree where the agent search an answer for an input, and for each piece of text its generating one work to answer to finally give a final output. These agents are very hard to train and tend to make grammatical mistakes. Some deep learning techniques are used in both models, like sequence to sequence, suited for generate text. Deep Neural Networks are powerful models are good for this kind of problems of speech recognition, and for an input it creates a set of outputs, and for those outputs it computes the probability of them with some formula to give a final answer. By now most of the systems are a retrieval-base model.

Also it's important to think which kind of conversations our agent should manage, they could be short, where it's aiming to output one single answer to a single input, and the long conversations, where it's important to keep a track of all that has been said.

Another important thing to consider is the domain of the agent, if it's closed, it will only respond to some questions of an specific topic. But if the domain it's open he must answer almost anything of any subject; a good technique for these systems is the use of hierarchical neural network models.

A recurrent neural network, receive an input as a sequence of tokens, these are translated in a vector and mapped to a dialog context, and start to make predictions of each token, and saves all the structures that we're made to keep track of all the information along the time.

Also a hierarchical recurrent encoder-decoder is good for this kind of conversations, used for web queries. It predicts the next web query using the previously queries submitted by the user.

And finally, because we want to integrate a personality, the outputs generated should be consistent, and **speaker model** and **speaker-addressee model** are two techniques to achieve this. Which basically has a set of words and a group of speakers that are related in some way (age, likes, country, sex), and for each word, it will be given a weight of how often that word is used in a group of speakers. The **speaker-addressee model** works the same way, but it tries to predict how an speaker 'x' will answer to an 'y' speaker.

System Characteristics

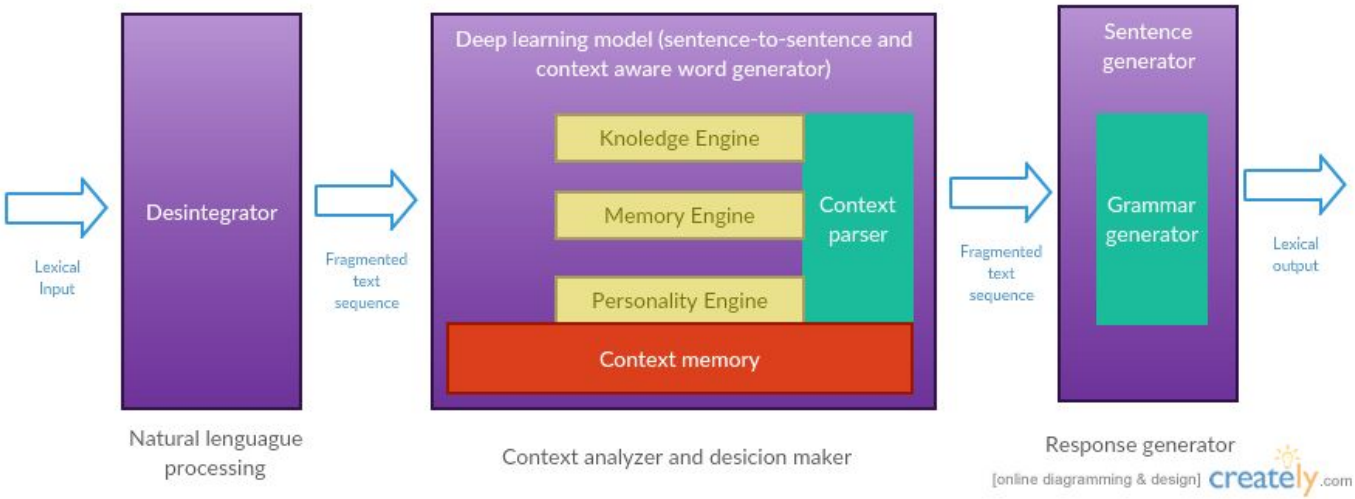
Before the appearance of deep learning a few year ago, all chatbots had hard coded, trying to predict what the user will say, and choosing an answer from a bunch of responses. So, we will create an End-To-End System, this means that our chat bot will use one system and it will be trained using one data see. The system will not make any assumptions about the use case or the structure of the dialog; the agent will be trained with the relevant data to be capable to chat with any person about that data.

Because conversations are a sequence of words instead of a fixed size input, we can't use a simple feedforward neural net; what we need is a neural net that accepts sequences as an input.

To achieve our goal we will use a recurrent neural net, in this net we feed the data back into the input while the agent is being trained in a recurrent loop.

Making use of Tensorflow, we will make use of an existence date set, to be able to create a model, this model will be trained with the assistance of Twitter users. As we don't want to make an evil and mean chat bot, the bot will only train its responses and won't learn from the conversations. But we want to keep training with new data sets our agent; so we will implement a way to have communication with the agent to correct their answers; and of course the developer will be in charge of these, avoiding the mean training from some users in Twitter.

Also, the TwinkerBot will have some memory to keep in touch with his friends in Twitter, saving information about previous conversations, like names, likes and and some important details about the users that talk with him. And finally it will have a personality module, with some fixed answers about the agent.



Experimentation

Taking an existence data set, in this case, the Cornell Movie-Dialogs Corpus, and start to train our model, using users from Twitter, so they will be sending tweets and waiting for a response. So at first, the responses may not have any sense, but the time it will give more precise answers. Also we will give some start information about the agent, to recreate a personality for him. Also we want to keep the training and learning so, it will implement some module to communicate with him and correct their responses, this module will only be available for developers as we don't want to train an evil agent. For the final step, the agent will keep track about the people who has talked to him, and record some previous information from previous conversations, giving information such as name, likes and any other interesting fact from his twitter friends.

To achieve this, we're gonna use Python 3.1 as our programming language and complement it with some modules to make possible the experiment.

The first module its "Tweepy", a library that will help us to use easily the Twitter Api, giving us the chance to search around tweets and create and response tweets. More than a module to communicate with Twitter, it's our way to train and keep the learning of our agent, also its window to communicate with the world. So all the tweets that the agent receives will be processed to train the agent and also to generate a response; for this step we will implement two important libraries. The first one is sqlite3, which give us the chance to use the SQLite databases to record information about the users who talk with the agent and remember some information about them. And the second library is TensorFlow, an open-source machine learning library developed by Google, and it will help us with the neural net implementation and combined with another library from the developers of TensorFlow; Seq2Seq, that will help us with the language processing and the creation of a response based on a data set and the training of our agent. After the process of the tweet, we will communicate again with our database to record some information if it's necessary and finally we will send a response to the user who wants to be friend of Twinkerbot.

Results and discussion

At the end we decided to train the bot using two different data sets, one using movie dialogs and the other using tweets. One of the first mistakes during the first attempt to train the model was a bad cleaning of the data set, so a good approach was to clean the set deleting all symbols like commas or quotes.

After 35000 to 45000 iterations, the loss value takes a lot to change, so we decided to stop the training process on that point. Also in the first attempt to train we don't split our data into training and testing sets, causing some overfitting on the results.

When we test the models, the responses are promising, and for each data set the answers are quite different. The twitter set don't respond to normal conversations, a very small percent of the people chat on twitter, because the network it's not for that, anyway it can talk about politics and bring too much love to his love. Meanwhile the cornell data set have more coherent responses related to the input of the user, and also more answers about daily life. We can see those answers in two different files on the project. The first one is on 'list.txt' with all the testing inputs for the twitter data set, and 'cornell.txt' for the movie dialogs data set. On each one we can see the progress of the bot during the time, having in one hand what the user says and what the result throwing by the trained model. Unfortunately all the answers don't have any notion of the context of the conversation and don't keep track on previous messages.

After a lot of iterations, the result are promising, anyway they could be better if we implement another way to clean the dataset, such as regular expressions; other could be have a computer with GPU that could help us with more powerful computations.

Also a way to support the mistakes of the bot is to training by ourselves, correcting the responses and saving it on a document. This document could be use to create our own data set. This method to correct it's implemented on the chatbot, learning how to fix the response, and could handle multiple options for a response, so the output will rarely be the same. With this same implementation some aspects of the personality of the bot were implemented. Another feature of the bot it's to learn about the twitter users, but right now this ability it's only limited to the name and likes of other users.

Finally the random interaction with the network was unable to implement, and related the activity with the personality could be a good work for the future.

Conclusion and future work

Making a really good chatbot it's a high level work, specially if you want to make to feel it like a human being; that means, to talk with other human beings, interact randomly with humans and also keep a track of what he learn of them and what he told and talk to them. Also the method in this chatbot was not trying to figure out what the human will say, so the output it's not always fixed, in this case we told the chatbot that we want to talk about a specific dataset.

So having a very good an extremely large data set could help to create a good chatbot, this data set must include information from different sources of conversation and interaction between humans.

In the future we will improve the chatbot interaction with the people on twitter, these interaction should be randomly and according to the personality of the bot. Also keep a track of the conversations to know what could be a good response for the context of the conversation. Also to improve the chatbot, the creation of the dataset based on the fixed responses and corrections made by the developers could give us an amazing chatbot, saving a lot of contributions by a lot of people in the world. Also the correction of the answers has some codes to prevent that anyone could train the bot and create a bad data set, a lot of mean responses.

Finally, the first approach in the future is retraining the dataset using another approach to clean the dataset using some regular expressions.

References:

[1] Perez D. & Pascual I.. (2011). Conversational Agents and Natural Language Interaction: Techniques and Effective Practices. United States of America: Information Science Reference.

[2] Hope R. (2016). Why Microsoft's 'Tay' AI bot went wrong. february 18, 2017, from Tech Republic Web Site: http://www.techrepublic.com/article/why-microsofts-tay-ai-bot-went-wrong/

[3] Sutskever I., Vinyals O. & Quoc L. (2014, december 14). Sequence to Sequence Learning with Neural Networks. Computation and Language, N/A, 9. 2017, february 18, from Cornell University Library Database.

[4] Serban I., Sordoni A., Bengio Y., Courville A. & Pineau J. (2015, july 17). Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. Cornell University Library, N/A, 8. 2017, february 18, From Cornell University Library Database.

[5] Li J., Galley M., Brockett C., Spithourakis G., Gao J. & Dolan B. (2016, march 19). A Persona-Based Neural Conversation Model. Cornell University Library, N/A, 10. 2017, 18 february, De Cornell University Library Database.

[6] Neubauer B. (2004, September). Designing artificial personalities using Jungian theory. Journal of Computing Sciences in Colleges, Volume 20, 9. 2017, 18 february, From ACM Digital Library Database.

[7] Cristian Danescu-Niculescu-Mizil. (2011). Cornell Movie--Dialogs Corpus. april 2, 2017, from Cornell University Sitio web: https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html

[8] Gelston L. (2017). Building a ChatBot, Pt. 2: Building a Conversational Tensorflow Model. april 2, 2017, by N/A Web Site: http://lauragelston.ghost.io/speakeasy-pt2/

[9] Huyen C. (2017). TensorFlow for Deep Learning Research. april 2, 2017, from Stanford Web site: http://web.stanford.edu/class/cs20si/lectures

[10] Shiv A & Bhalley R. (2016, november 21). ASR — A real-time speech recognition on portable devices. 2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA), 2, 4. 2017, april 2, from IEEE Xplore database.

[11] Ashangani K, Wickramasinghe K., De Silva D., Gamwara W., Nugaliyadde A. & Mallawarachchi Y. (2016, december 26). Semantic video search by automatic video annotation using TensorFlow. Manufacturing & Industrial Engineering Symposium (MIES), 1, 4. 2017, april 2, from IEEE Xplore database.

Attachments:

[1] Soucrce code: https://github.com/Jatapiaro/TwinkerBot_Revival