Final Project Report

Hannah Kerr

CS 401 Natural Language Processing
December 17, 2018

1 Introduction:

Chat bots are some of the most exciting and innovating applications of natural language processing and machine learning algorithms. Chat bots are computer programs that simulate and conduct human conversation. These programs have many uses and applications. Task based chat bots are intended to assist the user in completing a specific task or objective. Some common examples are bots that help users search for restaurants and hotels, make reservations for the user or answer questions a user might have. These implementation of these bots are primarily rule based and use pattern recognition and matching to converse with the user and complete the assigned task. Task-based chat bots typically have a limited scope of conversational abilities and struggle with any conversation outside of this scope.

Conversational chat bots differ from task based bots because their purpose is to recreate or mimic human intelligence and allow the user to have a full and realistic conversation with the bot. These chat bots typically involve more complicated natural language processing and machine learning algorithms. This report covers the creation and implementation of a basic conversational chat bot.

The result of this project is BotBuddy, a Markov Model based conversational chat bot. BotBuddy is trained on the Cornell Movie Dialog Corpus and uses Hidden Markov Model chains and other methodology to generate responses based on user input. Additionally, BotBuddy utilizes some regular expression pattern matching for further functionality, such as greetings,goodbyes, and knock knock jokes.

2 Related Works:

The creation and development of chat bots has been a hot topic in the field of machine learning for several decades and is only becoming more relevant. Therefore, there are many similar projects to this one that have been done. One of the main inspirations for this project comes from MegaHAL, a conversational chat bot created by Jason L. Hutchens and Micheal D. Alder [1]. MegaHAL was debuted at the Loebner Contest in 1998, an annual contest held to test artificial intelligence programs in their ability to display human-like intelligence. Hutchens and Alder created MegaHAL with the intent of creating a conversational chat bot that generates realistic responses to a users entry. To do this, Hutchens and Alder used Markov Modeling to create responses to return to the user. This method was very different from the rule and pattern based methods of previous chat

bots entered in the Loebner Competition, like ELIZA. The algorithm used by MegaHAL can be broken into the following steps [1]:

- Read the users input and segment into words and non-words (stop words).
- Find an array of keywords and use these keywords to generate several candidate responses.
- Display the response with the highest information to the user.
- Update the Markov models with the users input.

The methods and algorithm used by MegaHAL served as the primary inspiration for the methodology of this project.

Grzegorz Szymanski and Zygmunt Ciota explore how Hidden Markov Models can be used to generate text [2]. This paper focus's on generating words in Polish, using individual letters as the states within the Markov Model. Their findings indicate that HMM can efficiently be used to generate text based on the training text provided to the model. This can be expanded on to generate sentences rather than words by treating individual words as the states in the model rather than letters.

3 Methods:

In order to create and train the HMM algorithm, a large corpus of text is needed. As this chat bot is supposed to mimic human conversation, the Cornell Movie Dialog corpus was used as the training text. The Cornell Movie Dialog is a corpus of 220,579 conversational exchanges between 10,292 pairs of movie characters from 617 movies [3]. When downloaded the corpus comes as csv files each containing movie conversations, the lines, and movie metadata. All that is needed from this corpus is the actual dialog, so some pre-processing is required to extract and format the dialog lines into a separate text file. Since the Cornell Movie Corpus is a widely used data set, code to pre-process the data into the desired format can be found online. The code used in this project to extract and clean the data is based on the code provided in a chat bot tutorial using this same corpus by Matthew Inkawhich [4]. Only the code for cleaning and processing the corpus was used in this project as the rest of the tutorial covers using neural networks to build a chatbot, which is not the methodology being used in this project.

The primary 'brains' of the chat bot is the Markov Model generated on the corpus of text. HMM models a sequence of events where transition from one state to another depends on probability of the previous state. In this implementation, each state in the model is a word. This program creates a Markov object that creates a dictionary representing the Markov chains and uses the chains to generate text. When reading in the text file the object creates the matrix of possible state transitions and the frequency of each transition. This is stored as a dictionary. To allow the program to be run repeatedly and quickly, the dictionary created is written to a text file and can be loaded into the Markov object to be used in future calls. The function genDict() creates this dictionary based on the order specified and writes it to a text file. This function should only be run once as it takes a few minutes to generate the transition matrix and the file generated is very large. When a user initiates BotBuddy, the program first asks the user for their name. Whatever name the user inputs will be used to represent the user for the remainder of the execution. Word/pattern matching is used to identify if the user input is in a set list of common greetings or farewells. If a

the input matches one of these phrases then a rule based response is returned. For example if the user inputs a phrase that matches one of the predetermined greetings, BotBuddy will respond with a randomly chosen greeting response.

Regular expression matching is used to determine if the user entered the phrase "knock knock joke". If the pattern is matched, BotBuddy's knock knock joke function is triggered and BotBuddy will tell the user a random joke from a text file. The knock knock function also uses regular expressions to ensure that the user properly responds to the knock knock joke, telling them to try again if the input does not match the expected "who's there?" or "- who?".

If none of the rule or pattern based options are triggered, the program will use the Markov model to generate a response. First the users input is parsed and the parts of speech are tagged, using NLTK's POS tagger. Next the noun,verb, and pronoun are identified. One of which will be used as the seed for the Markov chain. If the user input does not contain any of these parts of speech, they are set to default to null and a random seed will be used. To determine what seed word to use, the program first tries to use the noun as a seed, if there was no noun in the input, the verb will be used. If there was no verb, then the pronoun will be used as the seed. Finally, if there was no pronoun, a random seed will be used.

The Markov object then will traverse the transition matrix to generate a sentence. There is a maximum sentence length to keep responses relatively short, for simplicity. Though the frequencies or weights of the transition states are calculated and stored in the dictionary, the next state is chosen randomly from the list of possible transitions in order to avoid repetition in the responses generated. Based on the seed given, two hundred possible responses are generated and returned.

These responses are then passed through a similarity measure. Using the scikit-learn and the code provided my Dr. Havill, a function computes tf-idf similarity of each of the generated responses with the user input. The purpose of this step is to select the response that is most similar to the users input based on the idea that the most similar response will likely make the most sense or resemble a possible acceptable response. Tf-idf stands for term frequency - inverse document frequency and it measures the number of times a word appears in a document and is offset by the number of documents in the corpus that contain the word. In this case, each of the possible candidate responses is considered a document. The cosine similarity to between the original user input and the tf-idf vectors is computed and the response with the highest cosine similarity is selected as the response and returned to the user. If there is no maximum value, as in some cases all of the responses have a cosine similarity of zero, a random response is selected from all of the candidates. This is done to continue the conversation and avoid the bot saying that it does not understand or has no response to give.

4 Evaluation:

There are a few ways to evaluate chat bots. The simpler form of evaluation is whether or not the bot achieves its task or purpose. The purpose of conversational chat bots is to mimic or recreate human conversation. This can be judged on how well the bot can generate reasonable responses. The ultimate test of conversational chat bots it the Turing Test.

The Turing Test was proposed by Alan Turing in 1950. It tests the ability of a machine to exhibit "intelligent behavior" that is indistinguishable from that of a human. If a chat bot is being tested against the Turing Test, a user would interact with it and not be able to tell if they were talking to a bot or another human. Since BotBuddy is a conversational chat bot, it could be subjected to the Turing Test, though BotBuddy would most definitely fail the test. From any interaction

with BotBuddy it is fairly obvious that it is a chat bot. Sometimes the responses that BotBuddy generates do make sense and could be realistic responses, but a majority of the time the response is some strange phrase that may vaguely relate to the input. Sometimes the response returned is just gibberish and makes so sense whatsoever. To date, no artificial intelligence program has passed the Turing Test, though many have come close.

5 Discussion:

BotBuddy is clearly a very simple implementation of a corpus based chat bot and as a result has a very limited conversational ability. There is a lot of room for improvement, both in this implementation and in the core methodology. The quality of the responses generated could be increased by adding a learning component to the Markov Model. Having the model learn based on the previous interactions could help create more realistic responses. Also, adding a form of memory, or way to track previous inputs could help create the context of a conversation. Right now BotBuddy only track the most recent input and uses that to generate the reply, but tracking the last two or three responses could help create a more coherent conversation.

The core 'brains' of the chat bot can also be improved. While Markov models can generate text that sometimes makes sense and is often entertaining, it is not the best method of producing logical and intelligible sentences. This is where more complicated machine learning algorithms would be a better option. Recurrent neural networks are commonly used in translating text and have been applied to text generation. Chat bots that use forms of neural nets can generate better and more coherent responses than Markov chain generation and are therefore a better choice if the goal is to create a conversational bot.

The result of this project is not a chat bot that can compete against more sophisticated algorithms or even attempt the Turing Test, but rather a bot that is entertaining and fun to talk to. Most of the responses do not make a lot of sense but sometimes a response will fit and can often be very funny. If the purpose of BotBuddy is to have a silly conversation and entertain the user, then the objective has been achieved.

References

[1] Jason L. Hutchens and Michael D. Alder. 1998. *Introducing MegaHAL.* In Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning (NeMLaP3/CoNLL '98). Association for Computational Linguistics, Stroudsburg, PA, USA, 271-274.

[2] Szymanski, Grzegorz Ciota, Zygmunt. (2018). Hidden Markov Models Suitable for Text Generation.

[3] Danescu-Niculescu-Mizil & Lillian Lee, *Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs.*, Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011

[4] Inkawhich, Matthew. "Chatbot Tutorial." PyTorch, 2017, pytorch.org/tutorials/beginner/chatbot_tutorial.html.