CS Capstone Technology Review

November 21, 2017

How to Make an Effective Robot Comedian

Prepared for

Oregon State University

Heather Knight

Prepared by

Group 13

AKA Robotics

Kevin Talik

Abstract

To make an effective robot comedian, one of the topics that is being researched is the dynamic performance set creation of jokes based from the audience reaction. Each joke that is told to the audience will illicit a reaction. If the robot can quantify heuristics for different types of responses, the robot can intelligently choose the next joke that fits the audiences preferences. By adapting to the unique qualities and preferences of different audiences, the robot can make a stronger attempt to connect to each person in the audience[1] [2]

Contents

1 Introduction

To make an effective robot comedian, we have developed three research questions that will be the basis of three internal systems for the machine. First, we are investigating how spontaneous interactions benefit a set. Second, we want to quantify the benefits of the robot's ability to personify it's character, and lastly how to adapt it's set corresponding to the audiences reaction. My role in this project is to develop an algorithm to adapt the robot's set of jokes based off of audience response.

Timing and anticipation for jokes are crucial for the success of the set. Every joke that is told provides information to the audience about the robot, and from the opening joke; each new part of the set will build the repertoire of the bot. If a joke is well received by the audience, the relationship between the comedian and the crowd is strengthened, as the people will become more empathetic towards the robot. Every joke will enable the audience to connote decisions, preferences and knowledge of the robot. This is where the connection, otherwise known as the ”Theory of Mind”, is made with the audience[1].

This paper is an overview for the technology that will be used to create this algorithm. There will be a brief discussion on some of the previous work done in this field. Next, there will be a comparison of programming languages that will best suit this project, followed by a section describing some of the available tools for creating an Artificially Intelligent machine. Since audience feedback is a large portion of how this robot will attribute heuristics of the audience, there will be one section describing different ways to provide input from the robot.

2 Individual Role in Project

Our group will be working collectively to make an effective robot comedian, but I will be specifically working on our third research question: the robots ability to adapt a set based off of audience response. From audience response to a joke, or ”bit”, the algorithm should be able to determine the best fit for the next joke. This algorithm will continue to find the next best fit for a set until the 3-6 minute set is complete, or the robot decides to that the audience is done listening to jokes. The portion of the algorithm that tests the audience's preference for humor will be known as the ”seed”. The seed will be a small subset of jokes, 2 - 3 jokes, that represent the collection of available jokes. For example, the seed bit may have three jokes; one joke could be a self-depreciating joke, one could be a joke about food, and another could be a quick observational joke about the audience. If the audience responded well to the self-depreciating joke, the algorithm should choose the next joke should be at the expense of the robot.

The seed of the set will give the audience pretense to the performance, and is the introduction of our robot. The delivery of the joke, and the content given has a large impact of robot character, and is out of the scope of the set creation; this is more suited toward the characterization research question. Also, the robot may need a more fluid way to interact with the audience (such as small talk, or crowd-work), which is underneath the breadth of the second research question about audience interaction. My contributions will be towards the system that determines which jokes, from our library of jokes, is best fit for our audience.

The algorithm will need to be able to input the strength of a delivered joke, and return a joke with attributes that match the strength of the joke. This will begin at the seed portion of the algorithm, and pick jokes until the set has lasted 3-6 minutes. The jokes will have some small variability in delivery, that correspond to the tasks of audience interaction and characterization.

3 Previous Work

There have been a couple of previous studies of robot theatre, most notably Dr. Heather Knight [3], Katevas et al [4], and Dr. Guy Hoffman [5]. To accomplish this task of designing an algorithm that can learn from the audiences' response to jokes, it is important to look at previous research, as well as the tools available for accomplishing this task.

3.1 ComedyParser

Katevas et al [4] has researched a robotic comedian agent previously with some success. During their research, they implemented a program called ComedyParser (https://github.com/minoskt/ComedyParser) that collects audience response information from SHORE computer vision, and performs the stand up set. The decision components, or what the robot does with information gathered from the SHORE vision, will be most important for implementing an algorithm in our project. A limitation with ComedyParser is that is specifically needs the SHORE vision to operate. SHORE will be too expensive for our project, and we will have to pursue more freely available systems. One solution that we had devised was to interpret the strength of the audience response to buttons on the robot. This will bypass the sensing component of comedy parser, as sensors for creating an audience model are out of the scope of this project. There is more discussion on on audience input in the "Feedback Methods" sections below.

3.2 Anticipation in Robot Theatre

Research conducted by Dr. Guy Hoffman studying the implications of anticipatory actions in social robotics [5]. This particular study found that humans working with a robot that can monitor anticipation for an event allows humans to anthropomorphize the robot with more human like attributes. This study uses non-atomic Markov Decision Processes (MDPs) to model the decisions for events. Additionally, Hoffman models anticipation with an impulse-cue situation, where the robot is waiting for an impulse to trigger a specific cue. This is non-deterministic, as the MDP process is modeled around the probability of an event happening.

4 Programming Languages

There are three pieces of software provided by the manufacturer that are required to program the robot: Choregraphe, NaoQi, and Monitor. Choregraphe is the What-you-See-Is-What-You-Get (WYSIWYG) software used to visually program tasks as "boxes", that can be linked together to create sequentially or simultaneously executed behaviors. Each behavior is a series of tasks (boxes in Choregraphe) for the robot consisting of dialogue, movements, and sensing. NaoQi is the name of the developers SDK for programming tasks to the robot over text. It comes in many programming languages such as C/C++, Java, and Python. Choregraphe has boxes that allow only Python scripts to be executed [6]. Monitor is the monitoring software that monitors the memory allocation on the robot, as well as the camera [7].

This section will discuss the benefits and downsides to each programming language that are available for this project. There are seven SDK variations that support different languages; the most complete versions of the NaoQI SDK are, in order from most complete to least, are C/C++, Python, Java [6].

4.1 Python

Python is a translated language that is popular in scientific programming [8]. Pythons simple syntax and simple library integration make it quick to prototype and implement systems. Python is the second most complete SDK in terms of

API functionality (C++ is the most complete) [9]. Python however is the only language that supports scripting from Choregraphe. Python is considerably slower than other compiled languages, and is not recommended for computations that are slower than 10 ms [6]. It is important to note that all of the C++ API functions are available from python as well [10]. Machine learning and A.I. are commonly implemented with Python, as such, there are a considerable amount of tools for implementing finite state machines and language processing. These tools are discussed below, in the "Libraries and Artificial Intelligence Tools" section.

4.2 Java

Java is one of the least complete SDKs that is available for NaoQi [6]. Java is an object-orientated compiled language that is commonly used in mobile and app development. It is a higher level programming language than C++, but lower level than Python. Java works well with JavaScript, which can be used to develop HTML applications for controlling the robot (known as QiMessaging) [11]. This could be useful if there is a problem with using sensors to receive input from the audience. There could be more information fields in a web application that may represent the audience better than audience testing, and could leave us further away from the robot while it is performing. Java does not have Choregraphe support.

4.3 C/C++

The C/C++ SDK of NaoQi is the most complete SDK [6]. C/C++ is an incredibly fast compiled programming language that has low-level programming qualities. Using C/C++ is required for loops that require faster than 10 ms response times [6]. The Aldebaran documentation specifies that this language is advised for advanced developers, and that Python be used before this. Additionally, this language does not have support in Choregraphe.

5 Libraries and Artificial Intelligence Tools

5.1 Natural Language ToolKit (NLTK)

For Natural language Processing (NLP), the Natural Language Tool Kit is one of the largest, and most complete Python library [12]. Other libraries, such as TextBlob, build on and improve some of the functionality in NLTK to more specific uses (Textblob is elaborated below). In NLP systems, the largest components are grammar parsing and the collection of text (known as *corpora*). NLTK comes with over 50 corpora for building a domain dictionary, as well as an API for constructing context free grammars (CFG). Context free grammars are a set of rules that generate sentences. However, this model is dependent on a string input, which could also be used to model more abstract state machines that the performance set consists of. This is a large library, with much more non-specific tools to develop language processing. More specific tools, such as Pykov and TextBlob are more mature for certain tasks, and may be more beneficial to use in modeling state machines or generating large sets of text, respectively.

5.2 PyKov: Markov Chains in Python

PyKov is a small Python module that is only for creating Markov Chains [13]. A Markov chain is a graph model with edges and nodes, where the edges are based on probability of traversing given edges. [14] is a simple visual explanation of how Markov Chains work. Markov Chains are different from CFGs in that a Markov Chain is dependent on the probability of traversing different trees through a graph; a CFG only accepts or rejects strings in a language, and may generate strings that are either incoherent or improbable. This is similar to how [5] models states in their robot that reacted to anticipatory events. Hoffman uses a Markov Decision Process, which is specific to an *agent* that interprets events and probability to traverse edges.

5.3 spaCy

SpaCy is a large library for language processing, that is an optimized library using C modules [15]. The benefits of this library is that it has deep-learning integration, as well as part-of-speech string tokenizing. It would be a stretch to use deep-learning for generating sentences because it would take the focus away from the algorithms purpose, which is to choose the next joke from observations of the audience, and heuristics of a joke. Deep-learning is a feature we do not need to use, but this library has large functionality for input language processing. This library's speed is a benefit, as Python is normally slower than most compiled languages such as C/C++. If we need a response time faster than 10 ms, it might be more beneficial to use this library over TextBlobs, which is not optimized with C modules.

5.4 TextBlob

TextBlob ”stands on the giant shoulders of NLTK” [16], but is specific to parsing ”textblobs”, and part-of-speech tagging. This library tokenizes text files, and creates dictionaries of seen words, as well as classifying words as nouns, verbs, adjectives, etc. NLTK can do this as well, but this module makes the task trivial. This could be useful in our project to specify word choice and building sentence data. However, it will be time consuming to manually find a corpus that conveys the same word choice and diction. Large corpora will be more difficult for our team to specify the importance of sentences and word choices. The other team mates are focusing on presenting a character from the robot, and smaller data could be easier for them to correlate their research goals to sentences.

6 Feedback Methods

6.1 Tactile Sensors on NAO Robot

There are a few touch sensors on the robot, two on each feet, two on each hand, and one on the head. This would be the most simple solution to implement, as each button would correlate to an audience response. For example, if the audience likes a joke, I would touch the left foot, and otherwise I could touch the right foot. A difficulty with this approach is that there is a limited amount of buttons, and giving the robot a small input may limit the information the algorithm can use to make a decision. There could be variations of buttons to represent different input, but touching the robot all over during a set may be distracting for an audience.

6.2 Audio Sensors on NAO Robot

Nao has 4 microphones on it's head, two on the front of the head, and two on the back. The sensitivity of each microphone is 20mV/Pa +/-3dB. This converts to -33.98 dBV/Pa. This means that a decibel noise that is 33.98 dBV/Pa when it is heard from the source will be picked up. The acceptable frequency range is 150Hz to 12kHz. This is a sensitive microphone, that would work better in small, controlled environments. The robot also can only pick up audio from one source at a time. Making this microphone work for a large audience would be difficult, and may be a challenge differentiating between a loud ”booing” sound, and a loud ”applause.” The benefit of using audio sensors is that we would not need to touch the robot while it is working, and could more naturally receive input.

6.3 Mobile Application for Operator Control and Audience Feedback

The Javascript module QiMessaging [11] is used to send messages to the robot over Socket.IO [17]. This would be useful to remotely send signals to the robot by using an HTML5 webpage. Using this model, we would bypass the sensors on the robot and directly communicate from it. We would use our observations of the audience to relay to the robot for the algorithm to make choices on the joke. This would be useful to have, but would be time consuming to create. Depending on the time frame of the project, we may not have time to implement a system like this.

7 Final Thoughts

For this project, I think that using Python is going to be a valuable choice. Modeling automata such as Markov Chains and Context Free Grammars in Python is going to be more productive than other languages, Java being a second choice for NLP. Additionally, there are a litany of Natural Language Processing libraries that are written for Python that will be helpful if use use a large amount of grammar choices. Javascript will also play an important role to receive input about the audience, and since we will be the only ones viewing this webpage, the work that would need to go into making QiMessaging function would be not as significant if we made it public. All of the libraries needed to make CFGs and Markov Chains have benefits, and can be used at the same time. NLTK is very complete, but some features are not as fleshed out, such as PyKov in making only Markov Chains. While the English language is an important role in our project, using audio sensors may become a challenge. It would be the most fluid for a performance if the audience interacted fully with the robot, but covering audience interaction with the microphones on the NAO robot may be a stretch goal.

REFERENCES

[1] Leslie, "Pretense and representation: The origins of theory of mind," *Psychological Review*, vol. 94, no. 4, pp. 412–426, 1987.

[2] Scassellati, "Theory of mind for a humanoid robot," *Autonomous Robots*, vol. 12, no. 1, pp. 13–24, 2002.

[3] Knight, "Eight lessons learned about non-verbal interactions through robot theater," in *International Conference on Social Robotics*. Springer, 2011, pp. 42–51.

[4] Katevas, Healey, and Harris, "Robot stand-up: engineering a comic performance," in *Proceedings of the Workshop on Humanoid Robots and Creativity at the IEEE-RAS International Conference on Humanoid Robots Humanoids (Madrid)*, 2014.

[5] Hoffman, "Anticipation in human-robot interaction." in *AAAI Spring Symposium: It's All in the Timing*, 2010.

[6] "Aldebaran documentation." [Online]. Available: http://doc.aldebaran.com/2-1/dev/programming_index.html

[7] "Aldebaran documentation." [Online]. Available: http://doc.aldebaran.com/2-1/software/monitor/index.html

[8] "Welcome to python.org." [Online]. Available: https://www.python.org/

[9] "Aldebaran documentation." [Online]. Available: http://doc.aldebaran.com/2-1/dev/programming_index.html

[10] "Python sdk - overview." [Online]. Available: http://doc.aldebaran.com/2-4/dev/python/intro_python.html

[11] "Aldebaran documentation." [Online]. Available: http://doc.aldebaran.com/2-1/dev/js/index.html

[12] "Natural language toolkit." [Online]. Available: http://www.nltk.org/

[13] Riccardoscalco, "riccardoscalco/pykov," Oct 2017. [Online]. Available: https://github.com/riccardoscalco/Pykov

[14] Vicapow, "Markov chains explained visually." [Online]. Available: http://setosa.io/ev/markov-chains/

[15] "spacy - industrial-strength natural language processing in python." [Online]. Available: https://spacy.io/

[16] "Textblob: Simplified text processing." [Online]. Available: http://textblob.readthedocs.io/en/dev/

[17] "Socket.io." [Online]. Available: https://socket.io/