CS CAPSTONE PROGRESS REPORT

FEBRUARY 16, 2018

HOW TO MAKE AN EFFECTIVE ROBOT COMEDIAN

PREPARED FOR

OREGON STATE UNIVERSITY

HEATHER KNIGHT

PREPARED BY

GROUP 13
AKA ROBOTICS

ARTHUR SHING
KEVIN TALIK
ANISH ASRANI

Abstract

To make an Effective Robot Comedian, we have developed three research questions to investigate our approach to designing this system: Adapting to audience response to humor, incorporating the crowd into the performance, and the effect of characterization in anthropomorphizing the robot. This paper outlines how our research questions will be incorporated into a system, and how we will study the shared space between a robot and an audience.

CONTENTS

1 Recap

A lot of the machines that surround us aren't very engaging to interact with. They serve their purpose, people get what they need, and the interaction is over. People do not consider robots as entities. That is the gap we are trying to close by performing stand-up comedy with a robot. Stand-up comedy is a casual and entertaining way for people to get more exposure to robots and see that robots are not just objects, but they are much more than that.

An effective robot comedian should be able to entertain the audience and generate laughs. We hypothesize that the effectiveness is dependent on three major aspects - crowd-work or the ability to integrate the audience in the performance, portraying a coherent and convincing character, and the ability to adapt the performance based on audience feedback. We will base our performances and studies around these three areas.

2 Adaptation (Kevin Talik)

This section of the report will cover the adaptation algorithm in the scope of this project. First, there will be a review of the goals in studying adaptation during a performance. The next section will summarize the current progress that has been made in implementing adaptive transitions. Concluding this subsection of this report will be a list of what next steps need to be taken, and current problems that are impeding the progress.

At a high level, the adaptation algorithm in this project will structure individual jokes into one performance set. The two goals of this result are: (1) to transition to topics dependent on the audience response, and (2) to present a crowd report upon completion of the set. The reason for presenting a crowd report back to the audience after the performance is that people may not notice actions the robot is taking to connect to the crowd.

2.1 Progress So Far

This term has been focusing on putting our individually programmed joke behaviors into one, cohesive program that will run the performance. The adaptation algorithm needs multiple jokes during the same performing cycle to compare what the audience like. We have animated eight jokes total, and have five that are currently in a single performance. Each joke that is physically animated in the robot will need a Python Object that describes the heuristics of the joke. Each joke object is appended to a list, and removed as they are performed. After the joke is told, the joke object will be pushed to a queue. The purpose of the queue is for the crowd report; each joke object will be popped off as they were told to construct the report of the audience. Below is a code example of the python joke object.

```
class joke(object):
    def __init__(self,name,ID,referenceSpace,topic,human=False,CW=False):
        self.qualities = {"name":name,
                          "ID":ID,
                          "referenceSpace":referenceSpace,
                          "topic":topic,
                          "human":human,
                          "CW:":CW}
        self.heuristics = []

    def addHeuristic(self,string):
        self.heuristics.append(string)


    def getHeuristics(self):
        return self.heuristics

    def getQualities(self):
        return self.qualities
```

Fig. 1. The joke object in Python.

User-defined classes can be added to the visual environment of *Choreographe* by editing the generated code of a visual block. The class needs to be appended (with additional library imports) at the top of the generated code.

The qualities dictionary member contains all of the essential information for tagging a joke. If there needs to be additional tags that are not in the dictionary, they can be appended to the *heuristics* list. Every joke that is told needs to be initialized before the performance. This figure shows the process of appending jokes to a set list.

```
def onInput_onStart(self):
    #self.onStopped() #activate the output of the box

    pList = []
    #print "Running"
    j = joke(name="DialUpHuman",ID="#0001",referenceSpace="Crowd",topic="Dating",human=True,CW=False)
    j.addHeuristic("Testing")
    pList.append(j)
    #print j.getHeuristics()

    j = joke(name="DialUpRobot",ID="#0002",referenceSpace="Shared",topic="Dating",human=True,CW=False)
    pList.append(j)

    j = joke(name="BreakALegRobot",ID="#0003",referenceSpace="Robot",topic="Jobs",human=True,CW=False)
    pList.append(j)

    j = joke(name="CarbonDatingRobot",ID="#0004",referenceSpace="Shared",topic="Dating",human=True,CW=False)
    pList.append(j)


    self.logger.info("Loading pickled jokes")
    for thing in pList:
        self.logger.info("Joke" + str(thing.getQualities()) )
```

Fig. 2. A set of jokes in Python

Each joke is initialized as an object with the correct arguments specifying the nature of the joke. Some of the more familiar arguments are easy to recognize: the name arg specifies the name of the joke behavior to run, the ID is a shorthand representation of the name, and the topic distinguishes the basis of different jokes.

The other arguments, referenceSpace, human, and CW are used in differentiating the research goals our group is studying. The human category is used by Arthur to tell robot character jokes from human-character jokes, and the CW argument stands for Crowd Work, which will help Anish Identify different degrees of crowd word. The referenceSpace argument is limited to one of three categories; Shared, Robot, Crowd. These are used for adapting between types of jokes. This refers to substance of the joke, and how it connects the audience, the comedian, and the world around them. For example, imagine a robot and a human are watching a dog walk down the street. The both of them observing the dog would be a shared reference, the human observing the comedian look at the dog would be a Crowd reference, and the robot observing the human would be a Robot reference. These categories help distinguish the social space between the crowd and the robot, and depending on the reaction to a joke, may highlight a preference of reference.

Generating our custom performance list speeds up testing, as animations need to be tested manually. Before the comedian is ready for the spotlight, each animation needs to be safe for the robot; our comedian needs to be able to survive each set. This requires careful iterations of movements to make sure that it does not fall over.

Between developing and animating on the robot, I have also been writing a draft for the experiments that need to be run to test the effectiveness of adaptation. Three different types of adaptation needs to be tested: (1) Active Adaptation (2) Passive Adaptation, and finally (3) no adaptation. Active Adapting is different than passive in that the robot will use live information from the audience to change the performance. Passive adapting should feel less engaging, as the response is not taken into account during the performance, but transitional actions are still evident. No adaptation will have pausing, and no audience input between jokes.

To summarize the work done for the adaptation algorithm thus far, I have collected our current jokes and put them into a single program, created a Python Class object for identifying jokes, implemented two data structures for remembering known jokes, and jokes told during a performance. By next term, we should be ready to experiment with our implementations.

2.2 Left To Do

By the end of this term, we need to have a working performance to test varying levels of adaptation from a bare-bones Crowd Report. Assembling all of our jokes into a standardized joke format with working data structures is a massive step in the right direction.

The next step is correlating a response to every joke told, so that our comedian can start making inferences about the audience from the performance. Once the responses can be validated with the corresponding joke, the robot can create a report of the responses that were witnessed.

2.3 Problems Impeding Progress

The animations are easiest and fastest to develop in the software *Choreographe*, provided by *Softbank Robotics*. However, remembering the jokes told during show is implemented in Python because input and output between functions is impossible with self defined data structures. Additionally, the Python 2.7 interpreter runs locally on the robot, and is difficult to test software on. This makes the developing and testing process slow and not automated. For animations, as our performance size scales upwards with more jokes, testing the safety of each performance permutation will take a lot of time to validate. Also, since our robot is a specific and expensive piece of hardware, finding information on bugs is both difficult and limited.

3 ANISH ASRANI

This section will talk about the crowd-work that goes into a successful stand-up performance. These should involve spontaneous interactions with the audience to improve the quality of the performance. The sensors on the robot are vital to implement this crowd-work as they can be used to trigger functions that can lead to audience interaction.

3.1 Progress So Far

I have started to understand how the sensors work a lot better. More specifically, the sensors to track audio and for facial recognition. The robot can turn its head to whatever direction it can detect sound coming from. It can determine if there is a face in its view and it can differentiate the number of faces it can see at a given time. It can also follow a face as long as the face is moving at a reasonable pace. These aspects will be integrated into the performance to make appropriate comments when it detects faces in the audience.

There is some speech recognition as well. However, that is not completely reliable as the noise levels get higher. That is why, it is important to stick to simple ”yes/no” questions to implement this in a real performance.

3.2 Left To Do

While we have managed to execute many jokes on the robot and different sensor interactions, we have to determine a solid script that connects all the jokes and interactions together. The jokes interactions should feel fluid and not sudden. They should not break the flow of the performance.

We also need to implement a block to determine audio levels to compare them throughout the performance. Higher volume levels could be construed as a positive response and that information can be used to determine future joke picks.

Once we accomplish some of these things, we can go ahead and perform shows with the robot. This will enable us to actually perform the experiments and research which is the whole purpose of the project.

3.3 Problems

One of the problems we have is that the sensors can be unpredictable at times. It can capture some ambient background noise and ignore the louder sounds which can lead to delays in triggering the relevant events on the robot. Sometimes, the the audio tracking sensors can lead to the ongoing animations to stop completely which can break the flow of the performance. We need to experiment more with how the sensors (like sound and face tracking) interact with the various animations and the robot animation frames in order to ensure that we can execute a successful performance without major interruptions on the robot. We have to limit ourselves to very simple questions due to the speech recognition not being very reliable for more complex responses.

3.4 Experimental Design

The general idea for this section is to test whether interacting with the audience (a.k.a crowd-work) throughout the performance, will improve the audience’s overall enjoyment of the show. We want to analyze the importance of this crowd-work relating to the central design of the project. Crowd-work can be done in different ways - calling out and talking to the audience, watching the audience and incorporating them into the jokes, and asking them questions to keep them engaged, or to build off to make new jokes.

There are various kinds of crowd-work we want to test: one research question is whether crowd-work matters at all, and the second is does the crowd-work needs to be real or robot can just pretend it is paying attention?

To answer these questions we suggest three research conditions: (1) no crowd-work, (2) fake crowd-work, (3) real crowd-work. The first one would be no crowd-work whatsoever. The robot goes about performing its set and does not directly address the audience at all. The second could spanover-the-top and inaccurate crowd-work, or best-guess crowd-work, with the possibility of being real (e.g., predicting that most people in the audience were from Oregon, even if it didn't really hear what they said). The third case would integrate actual robot sensing. It would be important for conditions #2 and #3 to be parallel to assess whether crowdwork really matters.

As condition one is fairly obvious, let us discuss deeper possibilities for condition #2. In the obviously fake research condition, the robot will talk to the audience directly but it will be completely wrong in its observation. The absurdity of a robot trying to understand the audience and being completely off could be entertaining for the audience, or it may not connect with the audience at all. The exact reception of this sort of crowd-work is something we are trying to study.

The other version of condition #2 is realistic but premeditated. For example, pre-known facts about the audience could be built into the robot or guessed. These pre-known facts could include the location of the performance, age demographics of the audience. For example, if the audience is known to be college-aged, the robot could be fed input to make comments about things relevant to college students.

Using actual robot sensing data is condition #3, and is certainly the ideal model, but requires sensing capabilities, processing power and hardware, so it would be good to know if it is really necessary. In this condition, the robot would be actually looking for cues from the audience during certain situations. For example, one example is asking questions and capturing words from the audience, then using that same word later. For example, the robot could ask a simple question about the weather, or the audience member's hometown. In this case, the robot can listen for specific words and ask another question about that specific town or city.

All of these conditions will be assessed with live audiences to check to what degree is crowd-work important for a robot comedian. The audience's response will be used to see if they enjoy a humanized robot or if they prefer a more robotic one, or maybe even a combination of both. As crowd work is just a form of human interaction, we expect it will improve the audience's perception of the robot's intelligence, add surprise to the show, and increase audience enjoyment levels. On the other hand, perhaps faking it can get 80% of the effect of the real version. That will be part of the evaluation.

4 Character (Arthur Shing)

4.1 Goal

The goal of this section of the project is to examine whether or not robot comedy can benefit from having jokes delivered from a robot's perspective. Our hypotheses are that a robot presenting jokes about technology or being a robot will be funnier than a human telling the same jokes, and that robots will be less funny than humans at telling jokes from a human perspective.

4.2 Progress So Far

So far, we have animated eight jokes. Of these eight animated jokes, three of them are working robot/human versions. This means we have five total unique jokes. In addition, we have three more written jokes that also have both human

and robot versions, but have yet to be animated. Besides these, there are also roughly a dozen other jokes that our client has written for us and are in review. The animated jokes are mostly "romance" related jokes, an intentional choice on our part. With jokes under the same topic, it would be quicker for us to come out with a whole working set. As it stands, at this point in the term we have under half of a full working set with robot and human versions. In terms of scripting, a

4.3 Left To Do

By the end of the project, we aim to complete three full working sets, with each set having 7-10 jokes. We have decided along with our client that it may be more expedient to instead create romance-related/job-related/age-related versions of each joke, as coming up with *good* jokes that can *have* logical robot/human versions is difficult and time consuming. While this may cause us to end up with less interesting set variety, it will still be adequate for the purposes of our project. By the end of this term, we should have at least one full working set. An example of a joke that has romance/job/age and robot/human versions is as follows:

```
I actually went on a date with a modem last week.
He was all –DIAL TONE, ACCOMPANIED WITH RAPID GESTURES.
I know he was just trying to connect
    but he s definitely too old for me
```

```
I actually went on a date with this guy last week.
He was all –BLAH BLAH WINE BLAH CHEESE BLAH. –
I know he was just trying to connect
    but he s definitely too old for me.
```

Whereas, the robot version of the above joke is shown below:

```
Hey, hey, I got news. This is big.
Ok, quiet down. Get this.
That's RIGHT folks.
I'm no longer single. *throws hands up*
I met a robot on tinder.
His name's Data. He's a really geeky robot.
Swiped right as fast as my motors could turn.
```

4.4 Issues

The main issue we have encountered in the project is time consumption. Animating jokes is incredibly time consuming, with each joke requiring on average 3-4 hours of animating and debugging, with their corresponding robot/human versions requiring another 1-2 hours of work. To animate the robot, things such as balance, logic of movement, vocal delivery pitch/speed, and timing need to be taken into account. In addition to this, there are usually issues that pop up during animation, such as motors overheating or an inability to connect with the robot, which occurs quite frequently. Actual walking is also often a problem, as it sometimes falls down. Another issue we have encountered is our collective lack of comedic genius. This became clear at the beginning of the term, when we realized how hard it was to create good jokes, and how it was even harder to come up with good jokes that have logical robot/human versions. To combat this issue, our client has been gracious with us and given her own input on jokes that could be used.

4.5 Experimental Design

For this portion of the project, testing will be done on Amazon Mechanical Turk. First, each joke in a topical set (romance, jobs, or aging) will be created with a corresponding robot/human version.

4.5.1 Methods

To address this goal, jokes will be written from a human or robot perspective. The jokes written from a human perspective will have a corresponding robot version, ideally with as much one-to-one correspondence as possible in regards to cadence, length of joke, parrallel content, similar motions, and so forth. These jokes will be subject to intense scrutiny by members of the project and by the client, such that revisions and edits can be made to create funny jokes with a definite correspondence between the two versions. For example, a human version of a joke might look like the following (lines with a definite correspondence with the robot version are highlighted):

```
1 Hey, hey, I got news. This is big.
2 Ok, quiet down. Get this.
3 That's RIGHT folks.
4 I'm no longer single. *throws hands up*
5 I met a man on tinder.
6 His name's Sebastian. He's a math nerd.
7 Swiped right as fast as my fingers could move.
```

Whereas, the robot version of the above joke is shown below:

```
1 Hey, hey, I got news. This is big.
2 Ok, quiet down. Get this.
3 That's RIGHT folks.
4 I'm no longer single. *throws hands up*
5 I met a robot on tinder.
6 His name's Data. He's a really geeky robot.
7 Swiped right as fast as my motors could turn.
```

4.5.2 Development process of joke writing

These jokes will be scripted in Choregraphe, where adjustments to vocal tones and pausing will be made. Then, animating the robot for non-verbal gestures will be done to enhance the delivery. The overall process may look similar to Figure 3.

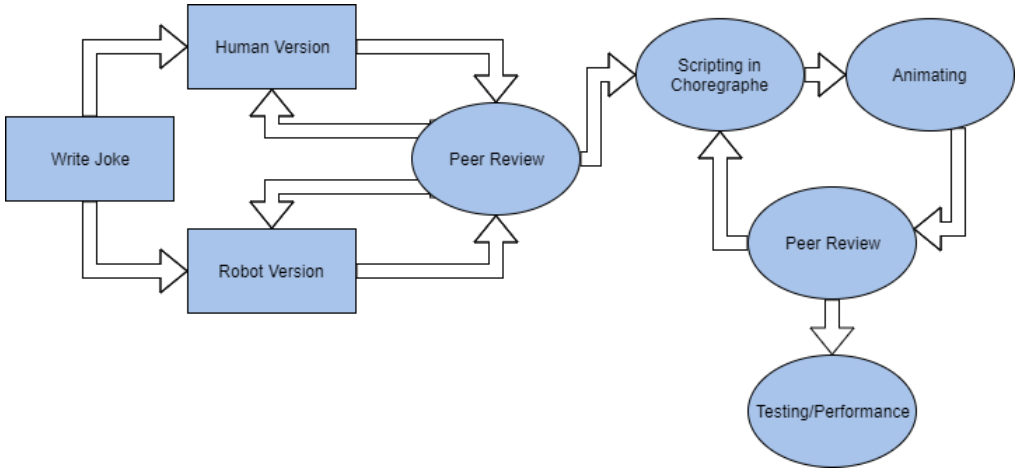

Fig. 3. The work flow from joke writing to testing.

4.5.3 Experimentation

The stand-up routine of the robot will comprise of text of the joke themselves, the motions that the robot uses to accompany them, and the way the robot surveys the audience after each punchline, e.g., in a human or robotic fashion. To determine the differences in audience response to the routines, studies will be done first on Amazon Mechanical Turk, and later with co-located audiences. Participants will be shown a video of the robot's stand-up routine, and then presented with a short survey. The routines will be between 5 and 10 minutes long, and the survey will include questions pertaining to each joke or routine. Participants will be compensated with standard rates for watching brief videos and answering survey questions.

5 CONCLUSION

Our project is coming together slowly, but surely. There are a lot of aspects we have accomplished successfully. It is just the matter of putting everything together and structuring the performances to make each one feel coherent and authentic.