Joke's On You: An Exercise in Joke Generation

Connor Ford and Gabe Magee

Pomona College
connor.ford@pomona.edu, gabe.magee@pomona.edu

May 3, 2019

Abstract

Joke generation is a difficult task for humans and machines alike. We consider a subclass of 'knock-knock' jokes to simplify the generative approach. Using an algorithm as opposed to training and testing more 'intelligently' we are able to create a small number of reasonable jokes with origins from movie scripts.

I. Introduction

There have been many successful approaches to joke generation Cai and Ehrhardt [1] tried to distinguish between a non-joke sentence and a joke one using Neural Nets. Yoshida et al [2] took in various image/caption pairs and tried to produce humorous captions given an image. Finally, Mihalcea and Strapparva [3] tried to apply Linguistic theories about humor to computational generation of one-liner jokes. These jokes typically follow a certain structure like call-and-response, or the more vulgar yo-mama. Others have trained models on large corpuses of data scraped from reddit or twitter. These have less associated structure and generally see more mixed results. We wanted to consider a less common joke-type in current literature: the 'knock-knock' joke. This joke type has a couple main advantages. (i) It follows a rigid structure. To illustrate the format, we annotate the following classic 'knock-knock' joke (not generated).

A: Knock knock.

B: Who's there?

A: Cash. [*Token*]

B: Cash who? [*Token + who = search word*]

A: No thanks, I'll have the peanuts [*Play on search word*]

(ii) There is a discrete set of 'knock-knock' joke subtypes [4]. One of which, as Taylor identifies, is word play on the *token*. We pursue this specific type because it always us to take a straightfoward, algorithmic approach to generation. The algorithm looks like:

(a) Generate *Token* from dictionary/corpus by identifying *search words* that end in the 'who'/'ew' and extracting the root of these words.

(b) Use corpus to generate responses based on *search word*.

II. Methods

i. Generating Tokens

Used corpus of $\sim 300k$ words and found words ending in the following characters: ["who",

[1] Cai, J., and Ehrhardt, N. (2013). Is This A Joke?.

[2] Kota Yoshida and Munetaka Minoguchi and Kenichiro Wani and Akio Nakamura and Hirokatsu Kataoka. (2018). Neural Joking Machine: Humorous image captioning.

[3] Mihalcea, R. and Strapparava, C. (2006). Learning to Laugh (Automatically): Computational Models for Humor Recognition.

[4] Taylor, J. (2004). Computationally recognizing wordplay in jokes.

"ew", "ewe", "ooo", "ooh", "oo", "hue", "ue", "eau", "eww", "hu"] Then we cross-referenced these words with a pronounciation dataset based on a CMU pronounciation dictionary ending in ["UW0", "UW1", "UW2"]. This only generated a list of 9 potential english words. To extend this, while sacrificing quality, instead of cross-referencing with the pronounciation library, we sacrificed with another english word dictionary. This outputs a list of about 68 english *search words* but the quality of the tokens took a large hit.

ii. Generating Responses

(i) Used bigram approach in Wikipedia dataset.
(ii) Used sentence following approach in movie script dataset.

III. Results

Name		
First name	Last Name	Grade
John	Doe	7.5
Richard	Miles	2

$$e = mc^2 \tag{1}$$

IV. Discussion

i. Subsection One

ii. Subsection Two

References

[Yoshida, Kota et al, 2018] Kota Yoshida and Munetaka Minoguchi and Kenichiro Wani and Akio Nakamura and Hirokatsu Kataoka. (2018). Neural Joking Machine: Humorous image captioning. *CoRR*, abs/1805.11850.

[Cai, J., and Ehrhardt, N., 2013] Cai, J., and Ehrhardt, N. (2013). Is This A Joke?.

[Mihalcea, R. and Strapparava, C, 2006] Mihalcea, R. and Strapparava, C. (2006), Learning to Laugh (Automatically): Computational Models for Humor Recognition. *Computational Intelligence*, 22: 126-142. doi:10.1111/j.1467-8640.2006.00278.x

[Taylor, 2004] Taylor, J. (2006), Computationally recognizing wordplay in jokes *Cognitive Science*