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 8, 2019

Abstract

Joke generation is a difficult task for humans and machines alike. It necessitates context, timing, and wordplay to elicit a laugh. We consider a subclass of 'knock-knock' jokes to simplify the generative and use a variety of straight-forward algorithms to create a small number of jokes with origins from movie scripts and the web.

I. Introduction

There have been many successful approaches to joke generation. [1] These jokes typically follow a certain rigid structure such as call-and-response. But others have trained more intelligent models on large corpora of data scraped from Reddit or Twitter. These have less associated structure and generally see more mixed results. We want to consider a less common joke-type in the current literature: the 'knock-knock' joke. Considering 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* $\approx$ *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 [2]. One of which, as Taylor identifies, is word play on the *token*. We pursue this specific type because it allows us to take a more straightforward, algorithmic approach to generation. The algorithm looks like:

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

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

II. Methods

i. Generating Tokens

We used a corpus of $\sim 300k$ words and found words ending in the following substrings: ["who", "ew", "ewe", "ooo", "ooh", "oo", "hue", "ue", "eau", "eww", "hu"]. We cross-referenced the *tokens* with another English language dictionary. This cross-referencing only generates a list of 9 potential english words. To extend this, while sacrificing a bit of quality, we cross-referenced tokens with a pronunciation dataset based on a CMU pronunciation dictionary with words ending in ["UW0", "UW1", "UW2"]. This outputs a list of about 68 english *search words*

[1] Cai, J., and Ehrhardt, N. (2013). Is This A Joke?. Yoshida K., et al. (2018). Neural Joking Machine: Humorous image captioning. Mihalcea, R. and Strapparava, C. (2006). Learning to Laugh (Automatically): Computational Models for Humor Recognition.

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

but the quality of the tokens took a hit. For example, it generates *tokens* such as 'sh' corresponding to *search word* 'shew'.

ii. Generating Responses

We decided to generate a response to a joke one of two ways:

(i) Using an n-gram approach learned from a Kaggle web dataset. This required stringing together phrases to artificially construct a response.

(ii) Using a 'sentence following' approach from a movie script dataset. We find the occurrence of a word in the movie script data set then return the next sentence in the dialogue or the next line of dialogue. This dataset is nice because movie scripts allow for a more conversational flow then most other corpora do. It also includes various colloquialisms making it better-suited for jokes.

III. Results

Below are a few hand selected jokes from four categories. The tokens either come from _cmu or _2d (where _2d is the list of 9 words and _cmu the list of 68 words referenced in section II.i). The responses come from either the n-gram method, the next dialogue movie script method, or the next sentence movie script method.

token	search word	response
		_cmu x n-gram
bl	blew	Blew out the candles.
val	value	Value of its currency.
		_cmu x next dialogue
curf	curfew	It won't happen again.
arg	argue	That's up to the judge, Gus. It's not your problem anymore.
		_2d x next dialogue
rev	revue	That job's gonna cost you a hundred bucks.
resid	residue	What does that mean?
		_2d x next sentence
tab	taboo	Set forth in the Sacred Scrolls.
mild	mildew	And is that it? More or less.

IV. Discussion

i. Why Human Evaluation

We decided to use human evaluation because of the nature of the results. It is more difficult to evaluate with traditional quantitative metrics. A score generated from collective human reactions to jokes generated may be a good way to evaluate our models.

ii. Evaluation Methods

To test our results, we surveyed just over 100 people each evaluating a random subset of 12 jokes (3 of each type). We asked them to score the humor of a joke and coherence of each joke on a scale of 1 (best) to 4 (worst). In the table below the results are listed as: humor (coherence).

iii. Evaluation Results

model	mean	sd	pct. 1
_cmu x n-gram	3.2 (3.1)	1.0 (1.0)	7 (7)
_cmu x next dialogue	3.1 (3.4)	1.1 (0.9)	13 (5)
_2d x next dialogue	2.9 (3.1)	1.1 (1.0)	15 (6)
_2d x next sentence	3.3 (3.5)	1.0 (0.8)	7 (2)

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*