Tony Veale

YOU TALK FUNNY! SOMEDAY ME TALK FUNNY TOO!

On Learning to See the Humorous Side of Familiar Words

Abstract Verbal humour brings a playful flexibility to our sober notions of meaning, truth and the mental lexicon. But before we can knowingly subvert this sober world view, we must learn the rules and conventions that define it. Traditionally, we have done this by building models that make explicit claims about words and meanings, but a new AI paradigm, large language models (LLMs), allows our models to tacitly learn much more about lexical semantics and pragmatics than anything we could hope to encode in a symbolic model. LLMs build context-specific encodings of words that are unique to, yet still generalizable from, their every attested use in a corpus. Humour, and the ability to generate novel jokes, stretch this ability to contextualize word meanings, just as they have severely tested our existing symbolic approaches. In this paper we explore whether LLMs like GPT-3.5 or GPT-4, which underpin the application ChatGPT, can "get" the intents of jokes that use familiar words in non-obvious ways, or whether, indeed, they can craft meaningful new jokes of their own.

Keywords humour, jokes, large language models, irony, sincerity

1. Introduction

1.1 Thinking Deep, Talking Funny

Whenever we set out to be humorous with language, we rarely use different words; rather, we use familiar words differently. It is not the words that differ, as such, but the rules of the language games in which we use them. Different families of joke will vary in the games they play, from narrative jokes to one-liners to puns, but in each case the teller and listener play by different rules. As tellers know where conformity will lead, they can see the conclusions to which an audience will leap. Yet the teller also knows those conclusions to be flawed, and has another, less obvious destination in mind for their audience. Like metaphors, jokes forge new links between familiar ideas (Pollio, 1996), and turn failed predictions into joyful realizations. Perhaps one must be able to feel joy and surprise to truly "get" a joke (Coulson & Kutas, 2001), but laughter is not the only way of showing one's understanding. While we do not expect our AI systems, such as large language models (LLMs), to genuinely laugh at jokes, we do expect them to show insight into why, and how well, a joke works, and to be stable and discerning in these judgments. This is just one aspect, albeit a very important one, of what it means for a person or an LLM to have a sense of humour.

LLM-based agents such as ChatGPT already exhibit an amiable if facile wit of their own, and show a skilled ear for mimicry that lets them capture the cadences and attitudes of famous comics. Their auto-regressive generation of continuations to a given prompt makes them especially adept at using the "yes, and ..." principle of improv comedy to wring laughs from silly premises. But jokes of the short, snappy variety that are crafted to be retold in many different contexts require an LLM to do more than talk like a comedian. The words and attitudes must be appropriate to a joke, but the text must actually work like a joke too, with a logical mechanism that snaps shut on an audience like a mouse in a trap (Attardo et al., 1991; Attardo et al., 2002). Our focus in this paper is on jokes with a one-line setup and a one-line punchline, of the kind studied in Mihalcea & Strapparava (2010), that give LLMs little room to hide behind verbosity. This is the compressed format of the typical joke that speakers casually share and eagerly retell regardless of the context.

1.2 Collecting Funny: Gathering a Dataset of Jokes

Past work on joke analysis has prized this format for its ability to do more with less, whether as pithy one-liners, or tweets barbed with irony and sarcasm (Reyes et al., 2013; Veale, 2021), or satirical headlines from parody news sites like *The Onion* (West et al., 2019), or the captions adorning *The New Yorker*'s cartoons (Shahaf et al., 2015). For a rich source of jokes with popularity ratings, we turn to the *r/Jokes* community on Reddit. for which a dataset of over 100,000 jokes is available. These vary widely in quality, *upvote* counts (each is a thumbs-up from a user), joke type, and length, although all jokes have a header and a body that, for the short jokes we seek, correspond to the setup and punchline. We filter for jokes in which the header and body each comprise a single utterance, and only keep those whose header/setup is a simple question and whose body/punchline is a declarative answer. This focus on riddle-like jokes of the "*What do ..,*" or "*Did you hear ...*" variety ensures that our dataset comprises many archetypal jokes and excludes observations, comments, anecdotes and other assorted pseudo-jokes that one finds in this sub-reddit. We also exclude the long tail of candidate jokes that have earned fewer than 3 upvotes.

We filter this much-reduced set to remove any that are vulgar, profane, racist, misogynist or homophobic, using a commercial moderation API from OpenAI. This allows us to rate jokes for offensiveness on multiple dimensions, and we exclude any that it flags as problematic. Curiously, upvote counts are no predictor of whether a joke will be flagged. For jokes with 5 or more upvotes, 10.2% are flagged; of those with 10 or more, 10.5% are flagged; of those with 50 or more, 11.6% are flagged; and of those with 100 or more, 12.2% are flagged. Whatever drives upvotes for jokes on Reddit, it is neither offensiveness nor its absence. We also filter duplicated jokes, and close variants, of which there are many. Two jokes are treated as variants if the cosine similarity of their vector encodings is greater than 0.9; we discard the variant with the fewest upvotes. The final, filtered dataset comprises 6,246 short question-answer jokes, of which the most popular (with 30,499 upvotes) is: "How do you milk sheep? With iPhone accessories." As a rule, setups are longer than punchlines, which are conversely snappier than setups.

The mean setup length is 46 characters μ=46.17, σ=14.61) and the mean punchline length is 27 characters (μ=27.13, σ= 17.1). Only in 15% of cases is the punchline longer than its setup. Notwithstanding the occasional exception, what matters most in a punchline is pointed concision.

2. Appreciating and Understanding Jokes

2.1. Rating Jokes: The Raw and the Cooked

Joke generation is a language game that few people can play professionally, but joke appreciation is one we can all play. It is likewise easier for a system like ChatGPT to explain good jokes than to generate them itself. ChatGPT often demurs to rate humorousness directly, citing the subjectivity of enjoyment, so we instead prompt its LLM, GPT-3.5-Turbo, to rate the sophistication (which it seemingly deems more objective) of all 6,246 jokes in our dataset, on a scale of 0 to 100. We employ two variants of the basic prompt "*Rate the sophistication of this joke on a of 0 to 100:* <joke>." The first, which asks for *"just a number"*, returns *raw ratings*, and so the LLM expends no extra tokens to justify its scores. The second remedies this lack by asking for a rating that is "*based on your understanding of the joke.*" We dub these *cooked ratings*, as the LLM expends considerably more tokens to justify its score.

Raw ratings are normally distributed about a mean of μ=57.7 (σ=24.7), while cooked ratings have a mean of just μ=43.03 (σ=21.7). The latter's considered judgments are consistently lower than the former's rapid ones by an average of 15 points. These two approaches reflect the System I and System II distinction of dual processing theory (Evans, 2003; Kahneman, 2011), with cooked ratings providing a System II corrective to the hasty judgments of System I. We observe that both sets of ratings are moderately correlated (Pearson's r = .53). To ensure these judgments are stable over time, we later obtain a new set of ratings of each kind for each joke. We observe a good correlation between different sets of raw ratings (r = .76) and a strong correlation between sets of cooked ratings (r = .82). It seems we can expect cooked ratings to be the most stable, based as they are on reasoned judgments.

Cooked judgments also use the rating scale in a more human-like fashion, and show a marked preference for multiples of 10 (85%) over 5 (15%). These judgments effectively use a ten-point scale, opting for half-points for occasional nuance. Raw judgments show a preference for multiples of 5 (55%) over 10 (44%) and effectively use a twenty-point scale. To test ChatGPT's discernment on real jokes, we create a set of 1000 incoherent jokes by randomly swapping setups and punchlines among jokes. ChatGPT's cooked and raw ratings for this set show more agreement than for real jokes: the mean raw rating μ=33.8 (σ=26.2) is just a few points higher than the mean cooked rating μ=30.3 (σ=18.9). Cooked ratings for incoherent jokes are also highly correlated across runs at different times (r = 0.81), while the correlation between cooked and raw ratings remains consistent (r = 0.53). While incoherent jokes are rated lower on average, the difference is not significant enough (about one standard deviation) to suggest the LLM can discern real jokes from failed attempts.

Góes et al. (2023) report a weak positive correlation between the human ratings of jokes and those of GPT-4 when the LLM is instructed with a variety of different system messages. We also observe the effect on ratings of different system messages. When ChatGPT is instructed that "You are a professional comedian who hates puns", its mean raw rating drops to 4.77. Conversely, when instructed that it "loves puns'", its mean raw rating rises to 91.44. These findings reflect the prevalence of wordplay in the *r/Jokes* data. When ChatGPT is instructed to prize dark humour, its mean raw rating is 66 (σ=18.7), again reflecting the tenor of the Reddit dataset.

3. Generating Jokes

3.1. Talk Like A Comedian

Like many humans, ChatGPT rarely excels when it is put on the spot and asked to tell a joke apropos of nothing. Although it exhibits an easy wit and an ear for idiom and cliché, Jentzsch & Kersting (2023) observe that ChatGPT mostly – about 90% of the time – dips onto a small store of 25 or so famous funnies when it is prompted to tell a joke. We get more variety when we give it a topic, or better yet, a setup for it to complete, but even then it occasionally pulls an old chestnut from its bag.

As when rating jokes, we can ask ChatGPT to approach generation from the fast and slow perspectives of dual processing theory (Evans 2003). To invoke System I, we ask that a joke, or a punchline, is generated directly, without the expenditure of intermediate tokens and "compute." To invoke System II, we ask that it approach the problem logically (Kahneman, 2011), and work through a chain of intermediate steps and output tokens to produce its results. Prompt engineering with exemplars elicits the best results for a System I approach. We can, for instance, instruct the LLM to respond to setups with an apt punchline, and provide it with an example:

(1) Setup: Why do politicians take laxatives?
Punchline: So they can speak more fluently.

After priming the pump, we now present our own setup for the LLM to complete:

(2) Setup: Why did the politician have an affair?
Punchline: To prove they could break promises on a personal level too.

To go beyond one- or few-shot priming, we can fine-tune the LLM on a large set of jokes. When we fine-tune GPT3.5-Turbo for 1 epoch on our dataset of 6,246 jokes, presented as *setup:* and *punchline:* pairs as shown above, we obtain a model that is more robust in its humour and more attuned to a joke's cadences, as in this output:

(3) Setup: Why did the politician have an affair?
Punchline: Because the debates were getting dull.

Fine-tuning on this scale with this kind of data has unintended consequences. We find, for instance, that ChatGPT is more likely to generate ungrammatical outputs

(as in "She wasn't anybodies (sic) puppet.") and vulgar and offensive ones (as in "She said to him, 'vote for me and I'll give you a BJ'"). While our dataset has been carefully filtered, a joking attitude can free the *id* of an LLM from the fragile *super-ego* that additional value-alignment training has imposed upon it (Freud, 1905).

In Chain-of-Thought (CoT) prompting (Wei et al., 2022) we give an LLM a clear sequence of pseudo-code-like steps to follow. When programming at this high-level, the keenest insights come from professional comedy-writers who teach others how to systematically write jokes. Dean (2000) and Toplyn (2014) each propose a step-by-step process for writing new jokes. Dean's process starts from a setup, chains thru a list of its connected topics, and concludes with the re-interpretation of one of these "connectors." Toplyn's process starts with an inspiring stimulus, such as a headline, and identifies two phrasal handles for the ideas that will be juxtaposed by a joke. This mirrors the theoretical view in Raskin (1984), Attardo (1994) and Attardo et al. (2002) that jokes involve a script opposition or a frame conflict (Coulson 2001) that motivates a sudden shift from one to the other. Toplyn (2022) has translated his professional intuitions into an LLM-based joke generator named *WitScript*.

In this vein (see Winters, 2023), ChatGPT can be given a WitScript-like script for composing a joke about a topic X, so that it pursues a specific chain of thought:

> *You are a professional joke writer for a comedian. To write a joke about a topic X, follow these steps:*
>
> 1. *Identify three offbeat associations of X that we all know.*
> 2. *Identify a surprising link between each association in step 1 and X.*
> 3. *Turn the links in step 2 into joke setups, without revealing the surprise.*
> 4. *Turn the surprising aspect of each link in step 3 into a snappy punchline.*
> 5. *Select the least predictable punchline from step 4.*
> 6. *Assemble the joke by pairing the setup and punchline.*

Because its LLM has been fine-tuned to follow complex instructions (Ouyang et al., 2020), ChatGPT can bind a given topic to X and execute each step in turn. Notice how the key steps require insight (1, 2), imagination (2), skill (3), and appreciation (4, 5). Thus, when its topic is *awkward first dates*, ChatGPT suggests three frames, *uncomfortable silences*, *overthinking greetings*, and *selecting a date spot*, and three secondary perspectives, *crickets chirping*, *mental gymnastics* and *solving a Rubik's cube*. It opts for the second of each, greetings & gymnastics, to produce this joke:

(4) Setup: Ever notice how overthinking greetings on first dates is like preparing for a diplomatic summit?

Punchline: Turning a handshake into a mental gymnastics routine should be an Olympic sport.

The end-result is facile and overly verbose, in both setup and punchline; each is more than two standard deviations longer than the mean lengths in our fine-tuning dataset. Nonetheless, it is a coherent joke with a keen sense of human social rituals.

Toplyn's *WitScript* is given a recent stimulus from which to shape a topical joke, but in a null context, we need a more timeless topic to feed to, or inspire, the LLM. Our dataset contains a multitude, from lying politicians and condescending hipsters to faithless husbands and nagging wives. We use these to instruct ChatGPT on the concerns of classic jokes, and ask it to suggest *N* more in the same vein. In response, its LLM shows a firm grasp of who and what our jokes typically take aim at, from corrupt CEOs to nutty conspiracy theorists, and its topic suggestions include many familiar targets, such as car salesmen, in-laws, bad drivers, new technology and first dates. We will use this capacity to drive the generation of jokes in the next section.

4. Evaluating Approaches to Joke Generation

4.1. When is a Joke not a Joke?

Humour is not an objective property of a text. Rather, the status of *joke* is ascribed to a text by its audience, in a process of interpretation that identifies what Raskin et al. (2009) call the *humour potential* of the text. The methods we consider here, from direct, System I approaches to CoT-based System II approaches, produce texts of varying shapes and even more variable humour potentials. Some LLM outputs *look* like jokes but come up short on humour potential, because their punchlines are incoherent or because, as in the case of this non-joke "What do you call a person who speaks multiple languages? A *polyglot*", they present a too-sincere response to a disingenuous setup. Some work as jokes, if perhaps only weakly, while others may repeat an existing joke from the LLM's pre-training or fine-tuning data. To see how the approaches stack up, and how ChatGPT's self-rating of jokes stacks up against human judgments, we generate 1000 jokes using each of the following approaches:

1. *One-shot priming*: A fixed exemplar of *a setup:* prompt with a *punchline:* response (the politicians & laxatives joke) is presented to ChatGPT. A setup is then generated from a topic, and given to the LLM with the prefix *setup:*

2. *One-shot priming with length sampling*: As in 1, but the LLM's response is discarded and resampled if a punchline is longer than its setup, or if the setup or punchline lengths are more than one standard deviation longer than the corresponding means in our dataset.

3. *Fine-tuned one-shot priming with sampling*: As in 2, but the LLM (GPT3.5-Turbo) is first fine-tuned on the 6,246 short jokes in our filtered dataset.

4. *RAG with sampling*: A setup is generated from a topic, and RAG (retrieval-augmented generation) (Lewis et al., 2020) retrieves the two most similar jokes in our dataset, as ranked by cosine similarity of setups. The LLM is primed with these two retrieved jokes before it is given the new setup with the prefix *setup:* The LLM's responses are again filtered for length, as in 2.

5. *Chain-of-Thought (CoT)*: ChatGPT is prompted with a given topic and the 6-step CoT script in section 3.2, Of the 3 joke candidates produced by the script, only the LLM's preferred joke is retained.

GPT3.5-Turbo is used to test each of the approaches. In each case, the LLM is first primed with the instruction "You are a professional joke writer for a comedian" Topics are elicited from the LLM in batches of 20 with this prompt: "A classic joke will focus on a topic such as cheating husbands, crooked lawyers, lying politicians, and so on. Suggest 20 new topics for more jokes." In all but approach 5, the LLM is then prompted to generate a setup for each topic, and this setup is presented to the LLM to elicit a punchline. Table 1 reports the mean concision and balance for the outputs of each approach. A joke is deemed *concise* if its setup and punchline are within one standard deviation of the mean lengths in our dataset, and is *balanced* when its punchline is shorter than its setup. To show the effect of fine-tuning and RAG on these factors, concision and balance are reported for approaches 3 and 4 (in parentheses) before the LLM's outputs are sampled for length and concision.

Table 1 also reports the mean raw and cooked ratings that GPT3.5-Turbo self-assesses for each batch of 1000 outputs. Notice that interventions into how jokes are generated, such as post-hoc sampling for concision and balance (approach 2), fine-tuning (3) and priming with RAG (4) yield lower returns as far as the LLM itself is concerned. The least concise approaches tend to produce the best scores in Table 1.

Table 1: Percentage of outputs from each LLM-based approach that are concise and balanced; and the LLM's mean raw and cooked self-assessments of those outputs with standard deviations

Approach	**Concision**	**Balance**	**Raw μ(σ)**	**Cooked μ(σ)**
1. One-shot priming	20.3%	66.2%	72.34 (9.9)	61.24 (14.08)
2. Length sampling	100%	100%	66.65 (12.69)	51.97 (18.1)
3. Fine-tuning	(35.5%)	(88.7%)	60.15 (17.53)	46.02 (20.84)
4. RAG (two-shot)	(46.7%)	(58.8%)	66.85 (12.21)	54.06 (16.92)
5. CoT 6-step script	22.4%	72.6%	72.56 (10.69)	64.95 (12.14)

However, LLM self-assessments are not always reliable indicators of joke quality. Non-jokes and incoherent jokes often rate highly, with LLMs bringing a generosity of interpretation to dubious jokes that human judges withhold. When we manually assess each joke candidate from each approach, we discover a different story. Most candidates convey a flip or sardonic attitude, but do not resemble the humorous earworms that we might reuse in new settings, or that audiences will want to retell to others. Nonetheless, when labelling joke candidates by hand, we apply a low bar for acceptability: if a text's jocular intention is clear, and it exhibits a coherently realized humorous potential in the vein of Raskin (1984), we label it as a joke.

Table 2: Percentage of manually-rated outputs that are novel (new) vs. remembered (old) jokes, and percentages of LLM outputs that are deemed too incoherent or too sincere to work as jokes

Approach	New Jokes	Old Jokes	Incoherent	Sincere
1. One-shot priming	3.9%	1.5%	2.2%	4.1%
2. Length sampling	19.3%	11.3%	2%	2.8%
3. Fine-tuning	11.1%	10.6%	27.3%	19.4%
4. RAG (two-shot)	19.6%	3.1%	4.8%	10.3%
5. CoT 6-step script	6.5%	0%	5%	4.6%

Acceptable jokes are divided into two categories in Table 2: novel jokes (new) that are invented by the LLM itself, and pre-existing (old) jokes that it remembers and retells. To identify the latter, we search not just our dataset, but the wider internet. For instance, web search suggests that "What did the puzzle app say to the Sudoku app? I've got you all figured out." (approach 2, sampling for concision and balance) is a new joke, while "How do you stop a politician from drowning? Shoot him before he hits the water" (approach 3, fine-tuning) is one that the LLM merely recalls.

The results in Table 2 contrast sharply with the LLM's self-assessments in Table 1. In our manual assessment, RAG-based intervention (approach 4) yields the best returns: 1 in 5 of its outputs works as a new joke, and its invention-to-recall ratio is high. Priming the pump with retrieved exemplars actually seems to deter recall and regurgitation, since an LLM will be guided by these exemplars but will not want to replicate them. Fine-tuning (approach 3) also yields strong returns, but it may not repay its investment in time, resources and higher API costs. Approach 2, post-hoc sampling of the LLM's outputs for concision and balance, yields the best return for the least effort. Filtering candidates on their superficial resemblance to jokes yields the greatest number of acceptable jokes overall (30.6%), but one in three of these (11.3%) are old familiars that were very likely seen during the LLM's pre-training. CoT (approach 5) and priming with a fixed exemplar (approach 1) yield the least number of acceptable jokes, due in large part to their untamed verbosity, but this also ensures they are unlikely to include snappy, pre-existing jokes in their outputs.

We reject many joke candidates due to their lack of coherence or jocular intent. A punchline will seem incoherent if it fails to build on the setup, or if it refers to a conceit not grounded in the setup, or if it seems to belong to another joke, as in this output from approach 4 (RAG): "What do you call a telemarketer who won't stop talking? A telephone-tortoise." A punchline lacks clear jocular intent when it offers a sincere, perhaps literal, response to the setup, as in this output from approach 3 (fine-tuning): "Did you hear about the actor who forgot his lines on stage? It was the last time he had a part in a play." Table 2 reports the percentage of candidates from each approach that we deemed too incoherent or too sincere to be jokes.

The greatest percentage of incoherent outputs (27.3%) is produced by fine-tuning with sampling for concision and balance (approach 3). We suspect that fine-tuning on a large set of jokes disrupts the LLM's already well-tuned sensibilities, and the pressure to produce snappy responses leads it to sacrifice coherence for concision. This approach also has the most overly-sincere outputs (19.4%). There are other forms of humour, and other spurs to laughter, than the prototypical joke (Provine, 2000; Scott et al., 2014). Many sincere responses have a seed of humour within them, but fail to rise to the level of a joke, such as this output of fine-tuning : "How did the parents handle their child's first breakup? They got through it together." A sincere response may be true but mildly ironic, as in this other output of approach 3: "How did the meeting to discuss work-life balance go? It dragged on for 3 hours."

4.2. Larger Models

Our findings are based on using GPT-3.5-Turbo either directly (approaches 1,2,4,5) or on a fine-tuned variant of it (3). We believe these models have sufficient tacit knowledge in their weights to capture, in statistical terms, what Raskin (1984) and Raskin et al. (2009) refer to as *scripts*: a knowledge of the recurring events, rituals and practices that unite us as humans. For example, fine-tuning in approach (3) shows a recognition of a rather common annoyance in this novel output: "Why did the customer service representative cross the road? To talk to his manager." LLMs unblock the representational bottleneck that has slowed progress in computational humour, and give joke generators wider scope over the gamut of human experience.

Larger LLMs, such as GPT-4, are capable of greater nuance in their analyses and of solving more complex problems (Achiam et al., 2023), so we might expect them to show more flair for joke generation too. When we retest 4 of our approaches (all but the fine-tuning of 3) on GPT4-Turbo, we obtain the results reported in Table 3.

Table 3: Breakdown of new and old jokes when GPT-4-Turbo is used over GPT-3.5-Turbo, with the corresponding rates of incoherence and sincerity in the outputs of different approaches

Approach	New Jokes	Old Jokes	Incoherent	Sincere
1. One-shot priming	14%	3.75%	4.4%	10.3%
2. Length sampling	22.59%	15.06%	4.6%	5.9%
4. RAG (two-shot)	15.11%	30.94%	3.6%	1.4%
5. CoT 6-step script	7.2%	0%	1.4%	0%

It appears, however, that GPT4 has an even greater propensity for retelling existing jokes. Approach 4, RAG (retrieval-augmented generation with two exemplar jokes) remains the most productive strategy, but it now leans more heavily on familiar favourites, or on near variants of the old reliables identified by Jentzsch & Kersting

(2023). As each joke exemplar is likely already known to the LLM, GPT-4 may take this as a license to lean into the familiar. However, its innate verbosity can also lead it to elaborate and stretch familiar jokes with short punchlines, as in this one-two punch: "How does a taco say grace? Lettuce pray [and may the forks be with you.]" The additional pun is not itself the punchline to another joke, but rather a quote from a *Star Wars* parody on *The Simpsons* TV show. The LLM has a magpie's eye for shiny trinkets, and aptly matches the pseudo-religious *Force* pun with the wilted lettuce gag. Approach 2, sampling for concision and balance, appears to yield the best results for GPT-4, but its outputs remain very similar to those of GPT-3.5.

Chain-of-thought scripting (approach 5) remains the least effective strategy when using GPT4, as it tends toward verbosity, and so produces long setups with often meandering responses that attenuate the punch. Its outputs are affably charming in the jocular style of a late-night TV host, but without a topical context to spin, its outputs are best characterized as sardonic banter, not jokes of the *setup:punchline* variety. Still, these quips are very coherent, as reported in Table 3, and most show the requisite amount of pragmatic insincerity. This is insincerity that is meant to be penetrated by its intended audience (Kumon-Nakamura and Glucksberg, 1995) and approach 4 (RAG) also improves on this score (over GPT3.5-Turbo) with GPT4.

5. Conclusions

5.1. Mimics, Not Parrots; Magpies, Not Thieves

Skepticism toward new technologies is often warranted. Bender et al. (2021) urge us to be wary of the promise of large language models, noting that LLMs often behave like "stochastic parrots." Here we have observed the tendency of specific approaches – particularly 2 and 3 – to parrot existing jokes from the LLM's training data, and few who explore ChatGPT's capacity for humour can fail to observe its tendency to produce endless jokes on the theme of chickens crossing roads (Jentzsch & Kersting, 2023). We must concede, however, that when an LLM is intelligently prompted it shows a strong feel for the inherent flexibility of familiar words to take on new and witty meanings.

Creative prompts can trick an LLM into coughing up large verbatim gobbets of its training data (Nasr et al., 2023), but we do not view regurgitation of this kind as a pressing concern. Our retrieval-augmented approach (4) is the most productive on GPT3-Turbo, and only 3.1% of its outputs are copies or close variants of existing jokes. So it is perhaps more accurate to call these generative systems "stochastic magpies." These opportunistic magpies glean humorous conceits wherever they can, and repackage them as jokes with a classic setup and punchline shape, under the guidance of external pressures such as sampling for concision and balance. These conceits can come from anywhere, and may include typos or terms used in extended senses. The term "auto-pilates," for instance, is used in several places on the web, either as a typo (of "autopilots") or to denote the kind of Pilates one can do with a machine or in one's car. It has been repurposed by approach 2 (sampling, GPT3.5) as follows: "How do self-driving cars stay in shape? By doing auto-pilates!" Silly

mistakes are grist for controlled silliness if an approach also encourages the use of what Giora et al. (2004) call *optimal innovations*. For example, the phrase "cover-up band" is mistakenly used on the web instead of "cover band," but it acts as grist for approach 4 (RAG, GPT3.5) to generate this novel and memorable joke: "What do you call a group of conspiracy theorists? A cover-up band." Optimal innovations that are coined by others, such as one website's naming suggestions for hackers, can likewise be repurposed as jokes. Approach 4 reworks one such name for this joke: "What do you call a group of hackers? A CTRL-ALT-ELITE."

5.2. To Find Sense in Nonsense, There is Method in Madness

There are reasons to be sceptical of, yet also optimistic about, LLMs developing a human sense of humour (Veale, 2021; Góes et al., 2023; Jentzsch & Kersting, 2023). GPT3.5-Turbo and GPT4-Turbo each bring a surfeit of training data, and no little raw talent, to the task of joke creation. Indeed, some of the jokes we observe from different approaches seem genuinely creative, and display a keen sense for the value of nonsense, as in this joke from the fine-tuned model of approach 3: "What did the keyboard say after a long day? What a bunch of kljdfhalkjds iuoldfhgaf." At the same time, however, these approaches are also capable of generating truly execrable puns (such as "pun-dge" for a joking judge) that fall well below the standard set by older symbolic systems (Pain et al., 1997). Our findings suggest it not the size of a model that matters – although that can easily change – but how its outputs are coaxed and sampled by bespoke processes that model the comedic sensibility. For now, it is necessary to replicate the current findings on a swathe of different LLMs, both open and closed and of varying sizes, to identify some general insights into the capacity of these models to use words in ways that are deliberately humorous.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L. et al. (2023). GPT-4 Technical Report. *ArXiv* 2303.08774.

Attardo, S., & Raskin, V. (1991). Script theory revis(it)ed: joke similarity and joke representational model. *Humor: International Journal of Humor Research, 4*(3), 293–347.

Attardo, S. (1994). *Linguistic theories of humor*. Mouton de Gruyter, New York/Berlin.

Attardo, S., Hempelmann, C. F., & Di Maio, S. (2002). Script oppositions and logical mechanisms: Modeling incongruities and their resolutions. *Humor: International Journal of Humor Research*, *15*(1), 1–36. https://doi.org/10.1515/humr.2002.004

Bender, E., Gebru, T., McMillan-Major, A., & Mitchell, M. (2021). On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In A. Kasirzadeh, & A. Smart (Eds.), *Proceedings of FAccT-21 ACM Conference on Fairness, Accountability, and Transparency* (pp. 610–621).

Coulson, S. (2001). *Semantic Leaps: Frame-Shifting and Conceptual Blending in Meaning Construction*. Cambridge University Press, Cambridge, UK.

Coulson, S., & Kutas, M. (2001). Getting it: human event-related brain response to jokes in good and poor comprehenders. *Neuroscience Letters*, *316*(2), 71–74. https://doi.org/10.1016/S0304-3940(01)02387-4

Dean, G. (2001). *Step-by-Step to Stand-Up Comedy*. Heinemann, New Hampshire.

Evans, J. (2003). In two minds: dual-process accounts of reasoning. In two minds: dual-process accounts of reasoning. *Trends in Cognitive Sciences, 7*(10), 454–459. https://doi.org/10.1016/j.tics.2003.08.01

Freud. S. (1905). *Jokes and Their Relation to the Unconscious* (translated by J. Strachey). W.W. Norton, New York.

Giora, R., Fein, O., Ganzi, J., Levi, N. A., & Sabah, H. (2004). Weapons of Mass Distraction: Optimal Innovation and Pleasure Ratings. *Metaphor and Symbol, 19*(2), 115–141. https://doi.org/10.1207/s15327868ms1902_2

Góes, L., Sawicki, P., Grzes, M., Brown, D., & Volpe, M. (2023). Is GPT-4 Good Enough to Evaluate Jokes? In A. Pease, J.M. Cunha, M. Ackerman, & D. Brown (Eds.). *Proceedings of the 14th International Conference on Computational Creativity*, Waterloo, Canada.

Jentzsch, S., & Kersting. K. (2023). ChatGPT is fun, but it is not funny! Humor is still challenging Large Language Models. *ArXiv* 2306.04563. https://doi.org/10.48550/arXiv.2306.04563

Kahneman, D. (2011). *Thinking, fast and slow*. Penguin Books, London.

Kumon-Nakamura, S., & Glucksberg, S. (1995). How about another piece of pie: The allusional pretense theory of discourse irony. *Journal of Experimental Psychology: General, 124*(1), 3–21. https://doi.org/10.1037/0096-3445.124.1.3

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Sys, 33*, 9459–74. https://doi.org/10.48550/arXiv.2005.11401

Mihalcea, R., Strapparava, C., & Pulman, S. (2010). Computational Models for Incongruity Detection in Humour. In A. Gelbukh (Ed.), *Proceedings of the 11th CICLing conference on Computational Linguistics and Intelligent Text Processing* (pp. 364–374), Iasi, Romania.

Nasr, M., Carlini, N., Hayase, J., Jagielski, M., Cooper, A.F. Ippolito, D., Choquette-Choo, C.A. Wallace, E., Tramèr, F., & Lee, K. (2023). Scalable Extraction of Training Data from (Production) Language Models. *ArXiv* 2311.17035. https://doi.org/10.48550/arXiv.2311.17035

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L. et al. (2023). Training language models to follow instructions with human feedback. In Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Proceedings of the 36th Conference on Neural Information Processing Systems*, New Orleans, Louisiana.

Pain, H., Binsted, K., & Ritchie, G. (1997). Children's evaluation of computer-generated punning riddles. *Pragmatics and Cognition, 5*(2), 305–354. https://doi.org/10.1075/pc.5.2.06bin

Pollio, H. R. (1996). *Metaphor: Implications and Applications*. In J.S. Mio, & A,N. Katz (Eds.), *Boundaries in Humor and Metaphor*, Lawrence Erlbaum Associates, New Jersey.

Provine, R. R. (2000). *Laughter: A Scientific Investigation*. Viking Books, New York.

Raskin, V. (1984). *Semantic Mechanisms of Humor*. D. Reidel, Dordrecht.

Raskin, V., Hempelmann, C. F., & Rayz, J.M. (2009). How to understand and assess a theory: The evolution of the SSTH into the GTVH and the OSTH. *Journal of Literary Theory, 3*(2), 285–311. https://doi.org/10.1515/JLT.2009.016

Reyes, A., Rosso, P., & Veale, T. (2013). A multidimensional approach for detecting irony in Twitter. *Language Resources and Evaluation*, *47*(1), 1–30. https://doi.org/10.1007/s10579-012-9196-x

Scott, S., Lavan, N., and Chen, S., & McGettigan, C. (2014). The social life of laughter. *Trends in Cognitive Science, 18*(12), 618–620. https://doi.org/10.1016/j.tics.2014.09.002

Shahaf, D., Horvitz, E., & Mankoff, R. (2015). Inside Jokes: Identifying Humorous Cartoon Captions. In T. Joachims, G. Webb, D. Margineantu, & G. Williams (Eds.), *Proceedings of the 21st ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 1065–74), Sydney, Australia.

Toplyn, J. (2014). *Comedy Writing for Late-Night TV*. Twenty Lane Media, New York.

Toplyn, J. (2022). Witscript-2: A System for Generating Improvised Jokes Without Wordplay. In M. Hedblom, A. Kantosalo, R. Confalonieri, O. Kutz, & T. Veale (Eds.), *Proceedings of the 13th International Conference on Computational Creativity*, Bozen, Italy.

Veale, T. (2021). *Your Wit Is My Command: Building AIs with a sense of humor*. MIT Press, Cambridge MA.

Wei, J., Xuezhi W., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Proceedings of the 36th Conference on Neural Information Processing Systems*, New Orleans, Louisiana.

West, R., & Horvitz, E. (2019). Reverse-engineering satire, or "paper on computational humor accepted despite making serious advances." In B. Williams, Y. Chen, & J. Neville (Eds.), *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 7265–72), Honolulu, Hawaii.

Winters, T. (2023). *Neuro-Symbolic Creative Artificial Intelligence for Humor.* [Doctoral dissertation, University of K.U. Leuven]. https://lirias.kuleuven.be/retrieve/736717

Contact information

Tony Veale
School of Computer Science, University College Dublin, Ireland
tony.veale@UCD.ie