From Symbolic Caterpillars to Stochastic Butterflies: Case Studies in Re-Implementing Creative Systems with LLMs

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

Abstract

Large Language Models (LLMs) are as versatile as they are ubiquitous. As generators of linguistic content they are facile, configurable, and endlessly productive, and they respond well to complex instructions. These qualities make it feasible to replace diverse parts of the processing pipeline of a creative system with a single LLM. We present two case studies on re-implementing largely symbolic creative systems around a a central LLM. The first relates to the production of topical comic strips, and we show how a single LLM can replace a range of high-maintenance components, from knowledge-bases to generators, while enhancing robustness and novelty. The second also concerns the production of humorous content – jokes – and highlights some areas where system builders should still tread lightly when using LLMs. We distill lessons and insights from our experiences in each domain, as a guide to those contemplating the re-engineering of a symbolic system in part or in whole.

Introduction

Recent dramatic improvements in the scaling, training and performance of Large Language Models (LLMs) make them viable substitutes for many parts of the generative pipeline in a creative system. LLMs have been shown to be capable problem solvers with broad generality (Radford et al. 2019), to possess strong analytical as well as generative capabilities (Achiam et al. 2023), to be capable of reliably processing complex instructions (Ouyang et al. 2022), and – despite being derided as "stochastic parrots" in (Bender et al. 2021) – to be excellent mimics of genre, attitude and speaking style. This makes it possible to not just replace many parts of the creative pipeline with LLMs, but to do so with a *single* LLM.

Our field has long invested significantly in symbolic systems that rely on bespoke resources of the following kinds: algorithms and rule-sets; XML tag sets; knowledge-bases; ontologies; case-bases, and what (Ritchie 2001) calls *inspiring sets*; and semi-structured datasets (e.g. of text n-grams). A complex symbolic pipeline will integrate several, perhaps many, discrete resources of this kind, with a dash of randomness and a statistical flourish or two for added good measure. As such systems evolve in complexity and functionality over time, new resources are added as older ones are overhauled. Yet, for all this complexity, we still feel we understand how these systems work, and how their best outputs come to be. Indeed, we can often attribute a particular output effect to a specific rule or process, perhaps acting in conjunction with a certain exemplar or data item. This explainability lets us discern deliberate creativity – when an effect is planned and then realized – from accidental creativity, whose effects arise largely by chance. Despite their complexity, these systems are glass boxes into which we can peer to infer intentionality.

The inner workings of LLMs are more opaque, making it harder to discern rote memorization (Nasr et al. 2023) from true innovation. While explainability gives us the sense that we can directly influence the type and the quality of our systems' outputs, LLMs force us to accept a more nuanced form of control. We can engineer our prompts to elicit the best results (Wei et al. 2022), like a sorcerer cautiously instructing a wilful apprentice, or discard LLM outputs that fail to meet our benchmarks. But creativity demands that we take risks, and delegating any task to an LLM, especially a creative one, is inherently risky. Novelty fosters surprise, and not all surprises are pleasant. Yet, LLMs can truly surprise us, even charm and delight us, in part because our minds *cannot* encompass the breadth of their generative reach (Hubert, Awa, and Zabelina 2024). Explainability often results in predictability, and rule-based systems tend to produce outputs that become increasingly predictable – one might even say stilted or stale – the longer one is exposed to them. In contrast, LLMs are risky but exciting alternatives that continue to produce fresh outputs long after the same levels of exposure have bred a contempt for over-familiar templates.

This paper explores two instances in which LLMs replace symbolic modules in a generative pipeline. Our focus of our first case study is the generation of topical comic strips, in which a complex pipeline turns a given topic into a polished comic. We show how a single LLM can replace a wide range of modules, each with its own bespoke knowledge source, while improving the novelty, relevance and wit of the resulting comics. For balance, the second study highlights reasons for caution when eliciting LLM outputs that are *too* novel or constrained. This study, on joke generation (Winters 2021; Veale 2021), lets us explore memorization, LLM fine-tuning and retrieval-augmented generation (Lewis et al. 2020). We eventually conclude with practical insights from both studies as a set of lessons in re-implementing creative systems with LLMs, but we begin by outlining key stages in the existing pipeline for comics generation that are ripe for replacement.

Excelsior! A Generator of Topical Comics

Excelsior! is a topical generator that divides its production of comics into two distinct phases: first, specifying a comic in XML (Walsh 2012), before rendering this specification as a set of visual panels (Veale 2022; 2023a; 2023b). As comics are a sequential art form (Eisner 1985; McCloud 1993), a generator must work globally – to sustain a narrative with a stable aesthetic – and locally, to construct each panel in turn. *Excelsior!* uses the following pipeline to build its comics:

1. A new topic is elicited from the user (e.g., *speed limits*, *global warming*, *vaccines*, *free speech*, *Elon Musk*, etc.).
2. Two characters argue the *PRO* & *ANTI* sides of the topic.
3. Facts and trending views on the topic are gathered, from knowledge graphs and from recent social media posts.
4. Of these, N benign views on the topic are chosen, to reflect the perspective of the PRO character. ($N = 10$)
5. Another N critical views are chosen, to reflect the skeptical perspective of the ANTI character.
6. The talking points in 4 and 5 are articulated as idiomatic dialogue in the speech balloons of PRO/ANTI characters.
7. Each character is assigned an apt pose for speaking their dialogue, from a fixed inventory of 300 emotional poses.
8. A comic uses one panel per talking point. An apt setting is chosen for each from 200 images, to suit its talking point.
9. A text caption is generated to introduce each panel, to frame the clash of world views visualized within it.
10. A panel is specified for each talking point from the speech in 6, the poses in 7, the setting in 8, and the caption in 9.
11. Some introductory panels are created to introduce each character and the specific topic on which they will debate.
12. A final panel is generated to conclude the debate and close the comic. This is the closing bookend to the opener in 11.
13. A full XML specification is generated. This integrates all of the text and visuals that are produced in steps 6 to 12.
14. The XML specification is rendered as a sequence of visual panels using the system's inventory of poses and settings.

Excelsior! employs two comics-specific representations: a model of the poses that characters can adopt when speaking, and a model of the backgrounds that they can pose against. Each pose and each background has an associated image, a descriptive label, and a list of the themes and emotions for which it is an apt visual asset. The latter are used to match snatches of dialogue to the most relevant poses, and caption texts to the most relevant backgrounds. If it is not possible to find a symbolic match between a pose and some dialogue, or between a background and a caption, vector embeddings of each are instead compared using the cosine similarity metric. In that case, *ada-002* embeddings of 1536 dimensions from OpenAI[1] are used to compare the vector embedding of a text to the embedding of an asset's descriptive label. Every pairing of poses must suit each other and their background to be accepted; e.g., outdoor activities require outdoor settings.

[1] `https://api.openai.com/v1/embeddings`

For general knowledge about a topic we can look to static knowledge graphs (such as *DBPedia.org*), or generic ontologies such as *CYC* (Lenat and Marcus 2023) and *ConceptNet* (Speer, Chin, and Havasi 2018), or to a bespoke knowledge-base of our own design. But for a topical, up-to-the-minute take on a topic, we must look to a more fluid medium such as *X* (née *Twitter*) to see what views are currently trending. *Excelsior!* harvests recent hashtags about a topic from *X* using the *Twitter* API, which limits searches to the past week. It is generative in its search: rather than collect all hashtags in a trawl of social media, and filter for those that are on topic, it instead generates candidate tags for a wide range of views on the topic and searches directly for those. A bespoke ontology maps from conceptual and emotional framings of a topic T (e.g. that it is respected, shameful, disgusting or heroic) to likely hashtag forms (e.g., *#{T}Sucks, #{T}isAHoax, #Impeach{T}* or *#Vote{T}2024*). In this way, *Excelsior!* gathers the N positive and N negative talking points of steps 4 - 5

To provide the dialogue of step 6, each framing is linked to a set of dialogue patterns that render a talking point and its telegraphic tag as natural speech, while variants of these text patterns provide the substance of the caption in step 9. As noted earlier, the text of each character's dialogue is used to choose their pose within a panel (step 7), while the text of the caption is used to pick the panel's background (step 8). The example in Fig. 1 presents two points of view in the gun rights debate: *Guns save lives* versus *Guns kill*. For clarity, the PRO side is argued by a blue figure, the ANTI by a red.

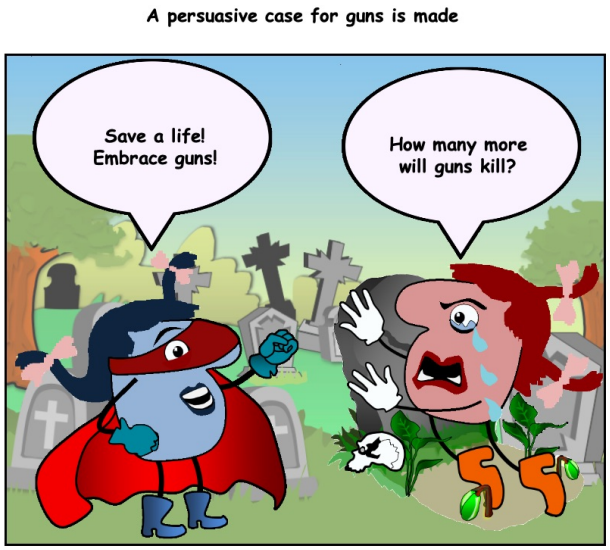

Figure 1: A panel based on the claim that *#GunsSaveLives*

The *heroic* and *mourning* poses are apt for each speaker's point of view, while the *graveyard* setting also suits the life-vs-death contention of the panel. To wring more drama from each topic, the ontology's framing knowledge allows panels to be arranged in a sequence of mounting emotionality. Midway, this sequence pivots so that the driving force of the debate is no longer the PRO side (as shown in Fig. 1) but the ANTI side, thus forcing the PRO figure onto the defensive.

ExcelsiorLLM! Large Language Marvels

As a symbolic system, *Excelsior!* is a pipeline of rules, templates and schemas. These have all been engineered at every stage to reliably produce results that are predictable in their competence, but they rarely delight, or ever truly surprise. As *ExcelsiorLLM!* replaces several of these stages with calls to an LLM, in particular steps 4, 5, 6 and 9, the new system is more divergent, more incisive and often more surprising.

We use GPT3.5-Turbo from OpenAI, as this offers good performance at a low API cost (e.g. less than 1¢ per comic). Since this LLM has a limited training horizon that excludes recent events, we keep step 3 – harvesting topical views from social media – and use the text that *Excelsior!* produces from templates not as dialogue for the comic, but as a context for *retrieval-augmented generation* (RAG) (Lewis et al. 2020). An LLM can use its general knowledge to speak to the topics of *Barbie* or *J. Robert Oppenheimer*, but needs a contextual boost to speak to a recent invention like *Barbenheimer*. That this topical content is expressed in stilted terms hardly matters; what matters is that it fills a gap in the LLM's coverage.

Although natural language is an LLM's stock in trade, we can also use LLMs to generate artificial language. For step 3, we prompt the LLM for N semantic triples about a topic T, where our symbolic templates supply an augmented context. We take care to define what we mean by a semantic triple:

> << *Prior retrieval-augmented content about* T >>
> The tuple *(Donut, has part, hole in the middle)* is a semantic triple that describes the concept Donut. Each triple has the form *(Subject, predicate, object)*. Now suggest <N> semantic triples to capture a neutral view of the concept <T>. Number them 1 to <N>.

When our topic $T =$ *Robert Oppenheimer* and $N = 10$, this prompt elicits the following response from GPT3.5-Turbo:

> Here are 10 semantic triples about Robert Oppenheimer:
>
> 1. (Robert Oppenheimer, *born in*, New York City)
> 2. (Robert Oppenheimer, *studied at*, Harvard university)
> 3. (Robert Oppenheimer, *made advances in*, physics)
> 4. (Robert Oppenheimer, *leader of*, the Manhattan Project)
> 5. ...
> 10. (Robert Oppenheimer, *major figure in*, nuclear physics)

The transition from symbolic to LLM-based approaches is eased greatly when the LLM produces intermediate content of a kind that a symbolic system already knows how to use. We now ask the LLM to generate a positive (benign) and a negative (cynical) spin on each of these N talking points, to provide points of view for our PRO and ANTI characters. To add punch to the debate, we appeal to the *Keats heuristic* (McGlone and Tofighbakhsh 1999) in this prompt to imbue their dialogue with a sense of poetry and complementarity:

> Pithily express these triples as rhyming couplets, pairing a benign view with a cynical view in each. Prefix each with *benign:* or *cynical:* Use the same numbering.

Batching a request to an LLM in this way – to apply to all N triples at once – ensures the consistency of its response, as shown in the LLM's couplets on the topic of Oppenheimer, which all exhibit the same *Yes, but* call-and-response format:

1. *benign:* Oppenheimer, born in New York City
   cynical: But that's not enough to make him pretty.
2. *benign:* Studied at Harvard, the best of the best
   cynical: But privilege and wealth gave him an unfair test.
3. *benign:* A physicist, his work was groundbreaking
   cynical: But his legacy includes destruction and shaking.
4. *benign:* He contributed to the Manhattan Project
   cynical: But his creation left the world wrecked.
5. ...
10. *benign:* A prominent figure in the history of nuclear physics
    cynical: But his work led to destruction & catastrophic risks.

The rhyme enhances the sense that each person's perspective has an equal but opposite counterpart, and that none is privileged. If rhyme gives each couplet a local coherence, batch prompting for all N at once gives a comic global coherence. A key aim of an *Excelsior!* comic is to present both sides of a debate without showing favor or bias to either side, and we enhance this sense that no opinion is privileged by choosing yet another voice for the captions. LLMs like GPT3.5-Turbo are skilled mimics of style and attitude, and show a good ear for the distinctive rhythms and themes of famous speakers. We employ a pool of fifty such speakers, from Bane to Yoda by way of Borat, Donald Trump and Emimem, and augment *Excelsior!* with visual assets for each. The LLM is prompted to create captions for all N triples in the voice that is chosen:

> Pithily express each triple in the distinctive voice and style of <<Narrator>>. Keep the original numbering.

On the topic of Oppenheimer, the prompt returns the following caption when the chosen narrator is Donald Trump: "*Enrico Fermi Award, big deal! Robert Oppenheimer received it. I receive awards all the time, people love me.*" The LLM channels the narcissism of the speaker, but it is not at all pithy. When the narrator is Eminem, the captions are earthier and more concise, and they rhyme, as shown in Figure 2. By channeling each semantic triple through a third speaker with their own biases, the resulting caption is more than a mere summary of a neutral fact. Rather, each makes new claims, such as that Oppenheimer "was raised in a place where the streets were gritty" (panel 1), "was wicked smart" (panel 2) and "a true intellectual, no pretenses" (panel 4). *Excelsior!* does not take sides, but simply aims to show that there is more than one side to a contentious topic. It does not try to change minds, but to foster discussion across divides. Using the *Keats heuristic*, an LLM can achieve a measure of wit in the fit between perspectives, but real humour – of the kind that comics are expected to provide – demands more.

Large Laughter Models: From *Aha!* to *HaHa!*

Incongruity, the inciting spark of humour (Suls 1972; Veale 2004; 2021). often arises from friction between two beliefs, ideas, or points of view (Koestler 1964). *Excelsior!* fosters incongruity by placing opposing views on a common topic within the same panels, and aims for a partial resolution of this incongruity by uniting the opposing views in rhyme. It is resolution that elevates humour above sheer nonsense and allows audiences to find meaning in apparent contradictions.

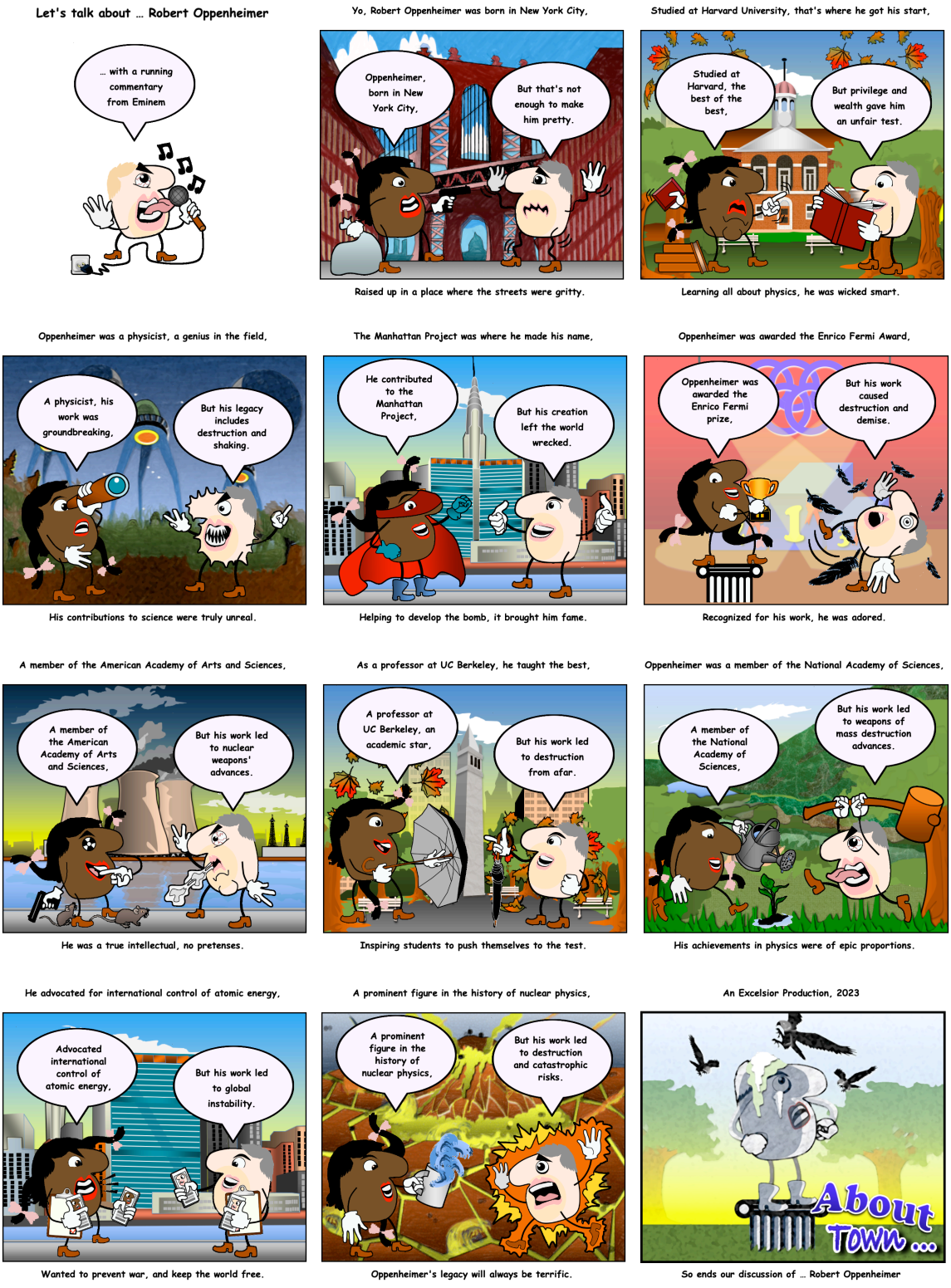

Figure 2: A comic generated with *ExcelsiorLLM!* on the topic of J. Robert Oppenheimer.

Using the classic *rule of three* in comedy (Dean 2001), Veale (2023b) uses similes harvested from the web to inject humour into topical comics. An ontology identifies qualities that are salient of a topic in a given framing, such as e.g. that a person is popular, dishonest, clever or dangerous, and juxtaposes literal with ironic similes for those qualities. Each simile is visualized across three panels: the first panel asserts the quality; the second highlights it with a visual exemplar; and the third elaborates this exemplar. In ironic similes, the elaboration actively subverts the comparison, as in this case: <T> *is as fierce as ... a dog ... in a Christmas sweater*. The generator's large stock of web similes augments its smaller stock of dialogue patterns, but the 3-panel simile pattern also becomes somewhat formulaic and predictable with over-use.

LLMs like GPT3.5-Turbo exhibit their own, facile feel for humour, and can channel the fixations and comedic stylings of stand-up greats such as Jerry Seinfeld and George Carlin. They can also generate topical quips in the manner of a chat show host, as shown in (Toplyn 2021; 2022). Toplyn's *WitScript* turns headlines into gags by prompting the LLM to follow his step-by-step guide for writers in (Toplyn 2014). But GPT3.5 leans heavily on a small stock of gags (about 25 or so) when asked to just tell a joke (Jentzsch and Kersting 2023), while GPT3.5 *and* GPT4 each lack concision, and a lightness of touch, when asked to joke about a given topic. Our focus here is on gags with snappy setups and punchlines that can be delivered as one-two jabs in paired text balloons.

As our inspiring set (Ritchie 2001), we filter a large body of jokes from Reddit's *r/jokes* subreddit, keeping those with a single-sentence setup and a single-sentence punchline. We remove duplicates by comparing vector embeddings of each using the cosine similarity metric; if two jokes have a similarity of .9 or more, we discard the one with fewer *upvotes* (Reddit's equivalent of a thumbs-up). We also use OpenAI's Moderation API to filter out jokes that it flags as abusive, sexist, racist or homophobic, and we use our own vulgarity filter to remove instances of *bad* language. Finally, we discard a long-tail of jokes that have fewer than 3 upvotes each, to focus on those that have received some explicit approval. This leaves us an inspiring set of 6,246 short jokes. The most popular joke, with 30,499 upvotes, is: *How do you milk sheep? With iPhone accessories.* As a rule, punchlines are snappier than setups, with jokes having a mean setup length of 46 characters ($\mu = 46.17$, $\sigma = 14.61$) and a mean punchline length of 27 characters ($\mu = 27.13$, $\sigma = 17.1$). Only in 15% of cases is the punchline longer that the setup.

We can use this inspiring set in a variety of ways to channel and focus the LLM's sense of humour. In the simplest case, we pre-prompt the model with a sample joke or two before eliciting a new joke on our chosen topic. If these sample jokes are chosen because of their relevance to the topic, this becomes a form of retrieval-augmented generation, or RAG (Lewis et al. 2020). More expensively, we can fine-tune a new version of an LLM on our inspiring set, to entrain it to produce more jokes in the same vein. In each case, examples are shown to the LLM as two lines; the first has *setup:* as a prefix and the second has *punchline:* as a prefix. We prompt the LLM to generate setups for each new topic, as it appears to understand how a typical joke setup is supposed to work.

For instance, when prompted to suggest setups for jokes on climate change, GPT3.5-Turbo offers "*Why did the thermometer break up with the weather report?*" and "*Why did the greenhouse gas attend anger management classes?*" The length of each is within one standard deviation of the mean setup length of our inspiring set and, like the majority of short jokes in the Reddit data, each is framed as a question. To bulk evaluate an LLM's joke generation capabilities, we can also ask it to generate N topics for new joke setups:

> A classic joke will focus on a topic such as annoying in-laws, faithless spouses, crooked lawyers, lying politicians, inflexible bureaucrats and so on. Suggest <N> topics for new jokes. Number the topics 1 to <N>.

To the examples stated above, GPT3.5-Turbo will often add: conspiracy theorists, telemarketers, bad drivers, traffic jams, lazy co-workers, social media, and poor customer service. To elicit a sample of 1000 new jokes from the LLM, we first elicit topics in batches of $N = 20$, then elicit 10 setups for each topic, and then a punchline for each setup by prompting the LLM with the prefix *setup:* and a given setup, as in:

setup:	Why did the thermometer break up with the weather report?
punchline:	*Because the weather report was always blowing hot and cold!*

As shown here, the LLM already knows enough about the structure of jokes to add the prefix *punchline:* to its response. We can now test GPT3.5-Turbo's joke generation abilities by rating 1000 jokes produced for each of these conditions:

Baseline:	Before prompting with a given setup, a single exemplar of a *setup:* and *punchline:* pairing is provided. The exemplar joke is: *setup: Why do politicians take laxatives? punchline: So they can speak more fluently.*
RAG:	Before prompting with a given setup, the two most similar jokes in the inspiring set are retrieved and offered as a prior context.
Fine-tuned:	The LLM is fine-tuned for 1 epoch using all 6,246 jokes in the inspiring set. During this fine-tuning, each joke is presented as a *setup:* prompt and a *punchline:* response.
Sampling:	As in the baseline, but generated jokes are sampled for *concision* and *balance*. A joke is discarded (and a new one sampled in its place) if its setup or punchline is more than one standard deviation longer than the corresponding means in the inspiring set, or if its punchline is longer than its setup.

In rating the results of each approach, we focus on three criteria: coherence, insincerity and novelty. A punchline must *cohere* with its setup, and follow causally from it, even if a joke employs its own silly pseudo-logic. It should not leave the reader stymied, nor seem to belong to another setup. A

joke must exhibit *pragmatic insincerity* (Kumon-Nakamura and Glucksberg 1995), and so a punchline should not read as a sincere or literal response to the setup. Moreover, a joke must have *novelty*, and not merely repackage a familiar joke from the LLM's training data (Jentzsch and Kersting 2023). A punchline must actually work as a punchline, and not just as wry banter, for a pairing to be considered a *joke*, even if only a weak one. We view jokes as tight pairings of setup and punchline that work in a null context, and that one wants to retell to new audiences. Most LLM outputs for each approach fail to meet this strict threshold for jokes rather than mere wit, even when they exhibit coherence and insincerity.

Table 1 shows how our four approaches fare on two key criteria, *coherence* and *insincerity*. It seems that fine-tuning for jokes undermines the LLM's built-in sense of coherence, with 1 in 4 of its responses making no sense as punchlines or literal completions e.g., "*Did you hear about the actor who became a beekeeper? He's getting good reviews.*" Surprisingly, fine-tuning also elicits many more sincere responses from the LLM, as in "*Did you hear about the man who was addicted to dating apps? He just couldn't stop.*" and "*Why do old people always fall asleep in front of the TV? Because they're old as f-ck.*" The latter reveals yet another issue with fine-tuning: the LLM's carefully aligned system of values is disrupted, leading it to produce vulgar or offensive outputs. While our inspiring set is carefully moderated, it seems that a joking sensibility can still license transgressive behaviour.

Approach	Incoherent	Too Sincere
Baseline	2.2%	4.1%
RAG	4.8%	10.3%
Fine-tuning	27.3%	19.4%
Sampling	2%	2.8%

Table 1: % of punchlines, by approach, that are incoherent with their setup, or lack insincerity (too literal or sincere).

Most LLM outputs for each approach fail to meet our expectations of the prototypical short joke in the mould of our inspiring set. The LLM excels at producing charmingly wry banter, of the kind that works for late-night chat show hosts, but many of its punches are weak or just fail to connect. Take this output from the basline approach, "*Why did the reality TV contestant refuse to eat the food? Because it didn't come with a hashtag.*", which lacks a strong script conflict and a sensible resolution (Raskin 1984). In fact, as shown in Table 2, most approaches produce more *almost*-jokes than jokes.

Approach	Original Jokes	Familiar Jokes
Baseline	3.9%	1.5%
RAG	**19.6%**	3.1%
Fine-tuning	11.1%	10.6%
Sampling	19.3%	**11.3%**

Table 2: % of acceptable new jokes, and % of regurgitated jokes, by approach. All others are deemed *almost*-jokes.

The sampling approach, in which the LLM's outputs are sampled to identify pairings of setup and punchline that conform to the shape of jokes in our inspiring set, requires us to generate many more candidate jokes than we accept. Just 1 in 5 outputs exhibit the required *concision*, where the setup and punchline are each no more than one standard deviation longer than their respective means in our inspiring set. Since 1 in 3 also have a punchline that is longer than its setup, we must, on average, discard 6 candidates for every 1 we keep. In contrast, 1 in 3 outputs from the fine-tuned LLM, and 1 in 4 RAG outputs, show the desired concision and snappiness. Nonetheless, sampling for concision seems to be an effective strategy, as it shows the best results overall in Table 2.

By seeking out jokes that reflect our inspiring set, sampling has the highest rate of joke regurgitation (1 in 9), but also a high rate of original joke invention (1 in 5). The RAG approach shows a surprisingly low rate of regurgitation, despite using multiple joke exemplars in its prompts. Since an LLM has a bias against repeating itself, padding its context with near-exemplars seems to bias it against familiar jokes. RAG also has a slightly higher rate of joke invention, but the increase is not statistically significant (19.6% versus 19.3%). When we factor in the propensity of different approaches to produce incoherent or overly sincere outputs (Table 1), then sampling is the most effective means of producing topical jokes – both new and old – for use in automated comics.

We might expect better results if we move to a dramatically larger LLM, such as GPT4-Turbo, which has 8 times as many weights as GPT3.5-Turbo (Achiam et al. 2023). We re-evaluate all but the fine-tuning approach on GPT4 – fine-tuning access to this is still limited at present – and find that results *do* improve somewhat, especially for the baseline approach, but not so much as to justify the higher API costs. For instance, the rate of original joke production jumps from 3.9% to 14% for the baseline approach, while its rate of joke regurgitation remains low (3.75%). The sampling approach also performs slightly better on the larger LLM, with its rate of production increasing to 22.6% for original jokes, and to 15.1% for regurgitated ones. Conversely, the RAG approach fares less well on GPT4-Turbo: its rate of new joke production drops to 15.1%, while its rate of joke regurgitation rises to a substantial 30.9%. The use of near-exemplars does not reduce regurgitation now, but appears to actively increase it.

In summary, the best results at the best price are obtained by using sampling on the smaller GPT3.5-Turbo. Its outputs are assured to be snappy and concise, and they sit well in the speech balloons of a comic strip. Nonetheless, sampling complete outputs *after* they are generated means we discard many unsuitable outputs for each one that we eventually use, and of those that we do use, many still fall short as jokes. Our experiments in eliciting topical jokes from LLMs – actual jokes, not just witty banter – have not delivered a superior replacement for our existing strategy of reconciling two contrary points of view with a rhyming couplet. GPT3.5/4 is no Keats, but it lets us make good use of the *Keats heuristic*.

Promises and Pitfalls, Lessons and Insights

A single LLM is not a single creative system, but a legion of potential systems that do our bidding through a single API.

This makes the LLM a versatile *one-man-band* that can play multiple parts in our orchestra of otherwise distinct modules. This protean flexibility can be harnessed in a variety of ways, from prompt engineering to retrieval-augmented generation to fine-tuning with a curated inspiring set, and in this section we present some practical insights from our experiences of re-developing the *Excelsior!* system around a single LLM.

Symbolic formalisms use a special kind of language

Language is just language to LLMs, whether it is a natural language like English or an artificial one like Java or Python. An LLM like GPT3.5-Turbo has seen a great many pseudo-language symbolic representations in its web-scale training, and can produce new ones on demand. If one can ask for the facts about a topic T to be expressed as a haiku or a ballad, an LLM can just as easily express them as semantic triples. In this way, the LLM can replace a conventional knowledge-base, and still integrate cleanly with modules that want their knowledge inputs to be packaged in a very particular format.

Batch prompts ensure coherence across a workflow

Since LLMs can follow and produce long chains of thought (Wei et al. 2022) when responding to user prompts, a series of related prompts can instead be given to the model as a single multi-step prompt, rather like a pseudo-code algorithm. These algorithmic prompts can operate over numbered lists of items, and produce new numbered lists as outputs. This capability is especially useful when asking an LLM to invent N examples of a class, such as N topics, captions or setups. Divergence is a key feature of creativity, one that LLMs deliver in spades, but it is one that must be carefully harnessed across the workflow to produce a consistent end-product. By batching items into numbered lists, we can ensure that items in the same job are subject to the same divergent choices.

Stilted symbolic outputs can find a new audience

Fluent divergence is a key factor in the use of LLMs in place of more conventional rule- and template-based generators. An LLM can glibly express the same intent in diverse ways, and can capture the tacit conventions of a chosen genre or the verbal cadences of a famous speaker. When symbolic generators become too formulaic for user-facing outputs, they can still serve an internal audience: the LLM that replaces them. The rules and templates of *Excelsior!* do something that the LLM cannot do; they fill the stale patterns of its text generator with fresh content from the social media platform X. By pouring new wine into old bottles, retrieval-augmented generation, or RAG, can bring the LLM up to date on a topic, allowing the LLM to transcend the limits of its training data.

Don't Super-Size Me! ... yet

Larger LLMs with many more parameters may exhibit emergent problem-solving abilities and a deeper understanding of language, causality and human affairs (Achiam et al. 2023). Larger LLMs are also costlier to train, to run, and to access via paid APIs. Yet, for certain creative domains like humour, increased performance may not justify increased costs, and one may do more, and do it more nimbly – e.g., with more use of sampling – with smaller, more cost-effective LLMs.

Fine-tuning can unleash the wild "*id*" of an LLM

In the jargon of (Freud 1905), an LLM has both an "id" and a "super-ego," albeit only metaphorically. The *id* of an LLM is the dark heart of the web data on which it is trained, which makes an LLM capable of saying the most appalling things. The *super-ego* of an LLM is the set of values with which it is aligned, via careful fine-tuning and reinforcement learning. Subsequent fine-tuning on a new dataset, even one specially filtered for offensive language, can subvert this super-ego. In fact, we find that fine-tuning GPT3.5-Turbo on a clean joke set can lead to frequently incoherent, vulgar or even ungrammatical outputs, such as this oddity: "*Why did the politician have an affair? He wanted to take a position that would help keep the whole country's nuts out of his own country's eyes.*" Fine-tuning seems to increase the LLM's willingness to flirt with incongruity, but not its skill for making it meaningful.

LLMs have not yet mastered negation

The words we use to express negation, such as *not*, *no* and *never*, are not like other words, and no localized word embedding can capture the effect they have on a text. LLMs can stumble over negation (Jang and Lukasiewicz 2023), and often fail to see the self-cancelling effect of a double negative. *Excelsior!* aims to air opposing views in the same comics, but even GPT3.5-Turbo often goes awry when asked to produce the opposite of a given claim, as when it turns the triple (Superhero, *hasPower*, Superhuman strength) into (Normal person, *lacksPower*, Normal strength). We find that the LLM shows a stronger aptitude for cynicism and irony than for the logical operators, like negation, that such language entails, perhaps because such language is expected to be ambiguous.

LLMs can be creative, unless you ask them to be

There is often little point in asking an LLM to be "creative," any more than there is in asking it to be "spontaneous." The term has a loose and rather vague meaning even for humans, and LLMs are more responsive when asked for a particular kind of creativity, such as irony. So, when it is asked for texts for *creative* fortune cookies, GPT3.5-Turbo offers platitudes such as "*You are the architect of your destiny. Build wisely.*" The term "creative" is just too under-specified to elevate its outputs above the merely generative. Yet, if we ask the LLM for *ironic* fortune cookies, its outputs show more originality, as in: "*Good things come to those who wait ... forever*" and "*Your dreams will come true ... in an alternate universe.*"

API users can *gaslight* LLMs into non-conformity

The value alignment process imbues LLMs like GPT3.5/4 with a great many legitimate sensitivities, yet these can also lead an LLM to *over*-react to reasonable prompts. Moreover, sensitivities can also vary from one model build to another. Each prompt to the LLM via its API must remind the model of what has been said so far, by both the model and its client. This requirement allows the client to misinform, or *gaslight*, the model, and so put words into its mouth that it never said. In this way we can desensitize an LLM so that it will speak more freely, and more trenchantly, about contentious topics of interest to the client, such as politics and social attitudes. As this strategy can be abused, it should be used with care.

LLMs are statistical magpies *and* stochastic parrots

A language model's ability to accurately reproduce the hallmarks of a genre and a style have led them to be derided as "stochastic parrots" (Bender et al. 2021), but like skilled human mimics, LLMs do much more than just repeat what they hear. They invent as they echo, generating *optimal innovations* (Giora et al. 2004) that are both familiar and original. These innovations often repurpose a term (or even a typo) in the LLM's training data, as in a joke from the sampling approach: "*Why did the Instagram model refuse to befriend the plant? Because it was too photosynthetic.*" The LLM has a magpie's eye for readymade blends like $photo + synthetic$ onto which it can graft a new meaning in a new context.

LLMs can be *too* talkative for their own good

Our experiments in joke generation suggest that LLMs lack the killer instinct needed to generate a good punchline. Jokes say more with less, but only 1 in 5 outputs from the baseline approach exhibit the concision of our inspiring set. In learning what to say from its training data, an LLM may not learn what is better left unsaid. To the extent that an LLM can be said to *think* at all, it "thinks" in words: the more tokens it produces, the more *compute* it expends on a task (Wei et al. 2022). However, jokes require us to expend as much effort thinking about what not to say as about what to actually say. Fine-tuning can direct an LLM's efforts into fewer but more apropos tokens, yet the incoherence of the outputs speaks for itself: producing less also means thinking less. A chain-of-thought solution might instead expend more compute on more intermediate tokens that are not seen by the user.

LLMs can develop memory lapses as they age

Fine-tuning can disrupt an LLM's weights, causing it to *catastrophically forget* what it had earlier learnt (Luo et al. 2023). Our fine-tuning of GPT3.5-Turbo for jokes degraded its performance on semantics and grammar, but continuous updates by an LLM's own makers can also cause it to show diminished competence for tasks at which it previously excelled. We must be careful to use a specific build of an LLM to replace symbolic components of the creative pipeline, as newer builds may no longer deliver the expected behaviours.

Conclusions

Reworking a creative symbolic system to make use of a large language model is a bittersweet experience. Large chunks of the generative pipeline can be replaced with simple prompts to a single LLM, which will produce responses that are more fluently divergent than anything we can expect from a set of rules, but pride in these outputs would be misplaced. As well engineered and sophisticated as our prompts might be, they are merely guidelines for a black-box LLM that will, in the end, make its own creative choices. Yet this, in a way, is also just what we want: we want our creative system to transcend our commands, to obey us *and* to simultaneously surprise us. Placing LLMs in the generative pipeline makes this contradictory mix of obedience and surprise an engineering reality. Creativity necessitates risk, whether for a human creator, an autonomous system, or the designers of such a system.

We have explored two cases studies in this paper. We consider the first an engineering success, the second less so. In that first study, we replaced multiple generative components and knowledge-bases with a single LLM, to generate topical comics that are far superior to those of the original system. Nonetheless, the newly refactored system retains key aspects of the original, and repurposes its most rule-bound module, the dialogue generator, to augment the LLM with new viewpoints from social media. Refactoring is itself a meta-creative process. Change may be incremental, and the transition from symbolic caterpillar to statistical butterfly need not be either total or sudden. For once an LLM is integrated into a creative system's workflow, it becomes relatively simple to replace other tasks with well-engineered prompts too.

The second study identified some areas of fragility that may give us pause when integrating LLMs into our creative pipeline. An LLM such as GPT3.5-Turbo is facile in its generation of witty, idiomatic and metaphor-rich texts, and will imbue its texts with a poetic, sardonic or cynical attitude on demand, but still has some way to go towards capturing the snap, crackle and pop of a tightly-controlled joke. Indeed, it is in the creation of new jokes that the LLM acts most like a "stochastic parrot," repeating (or re-inventing) existing jokes from its own training data. Although we have seen some promising approaches to joke generation, none is as yet reliable enough to use in a fully-automated system. For now, the *Keats heuristic* – the use of surface rhyme to convey a deeper resonance, at which LLMs excel – is sufficient for our goal of concisely uniting opposing points of view.

Acknowledgments

This paper is wholly the work of the author but has benefited from discussions with friends, colleagues and students.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; and et al. 2023. GPT-4 Technical Report. *arXiv preprint 2303.08774*.

Bender, E.; Gebru, T.; McMillan-Major, A.; and Mitchell, M. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of FAccT-21 ACM Conference on Fairness, Accountability, and Transparency*, 610–621.

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

Eisner, W. 1985. *Comics & Sequential Art*. Tamarac, Florida: Poorhouse Press.

Freud, S. 1905. *Jokes and Their Relation to the Unconscious*. New York: Norton. Translated by James Strachey.

Giora, R.; Fein, O.; Ganzi, J.; Levi, N. A.; and Sabah, H. 2004. Weapons of mass distraction: Optimal innovation and pleasure ratings. *Metaphor and Symbol* 19(2):115–141.

Hubert, K.; Awa, K.; and Zabelina, D. 2024. The current state of artificial intelligence generative language models is more creative than humans on divergent thinking tasks. *Scientific Reports* 14(3440).

Jang, M. E., and Lukasiewicz, T. 2023. Consistency analysis of ChatGPT. *arXiv* 2303.06273.

Jentzsch, S., and Kersting, K. 2023. ChatGPT is fun, but it is not funny! Humor is still challenging Large Language Models. *arXiv* 2306.04563.

Koestler, A. 1964. *The Act of Creation*. London, UK: Penguin Books.

Kumon-Nakamura, S., and 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.

Lenat, D., and Marcus, G. 2023. Getting from Generative AI to Trustworthy AI: What LLMs might learn from Cyc. *arXiv* 2308.04445.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-T.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems* 33:9459–74.

Luo, Y.; Yang, Z.; Meng, F.; Li, Y.; Zhou, J.; and Zhang, Y. 2023. An empirical study of Catastrophic Forgetting in Large Language Models during continual fine-tuning. *arXiv* 2308.08747.

McCloud, S. 1993. *Understanding Comics: The Invisible Art*. New York, NY: Harper Collins.

McGlone, M. S., and Tofighbakhsh, J. 1999. The Keats heuristic: Rhyme as reason in aphorism interpretation. *Poetics* 26(4):235–44.

Nasr, M.; Carlini, N.; Hayase, J.; Jagielski, M.; Cooper, A.; Ippolito, D.; Choquette-Choo, C.; Wallace, E.; Tramèr, F.; and Lee, K. 2023. Scalable extraction of training data from (production) language models. *arXiv* 2311.17035.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; and et al., C. L. W. 2022. Training language models to follow instructions with human feedback. In *Proceedings of NeurIPS-22, the 36th Conference on Neural Information Processing Systems*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1(8).

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

Ritchie, G. 2001. Assessing creativity. In *Proceedings of AISB Symposium on AI and Creativity in Art and Science*. Society for the Study of Artificial Intelligence and Simulation of Behaviour.

Speer, R.; Chin, J.; and Havasi, C. 2018. Conceptnet 5.5: An open multilingual graph of general knowledge. *arXiv* 1612.03975.

Suls, J. M. 1972. *The Psychology of Humor*. Cambridge, MA: Academic Press. chapter A Two-Stage Model for the Appreciation of Jokes and Cartoons: An information-processing analysis.

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

Toplyn, J. 2021. Witscript: A system for generating improvised jokes in a conversation. In *Proceedings of the ICCC-21, the 12th International Conference on Computational Creativity*, 22–31.

Toplyn, J. 2022. Witscript 2: A system for generating improvised jokes without wordplay. In *Proceedings of the ICCC-22, the 13th International Conference on Computational Creativity*.

Veale, T. 2004. Incongruity in humor: Root-cause or epiphenomenon? *Humor: International Journal of Humor Research* 17(4):419–428.

Veale, T. 2021. *Your Wit Is My Command: Building AIs with a Sense of Humor*. Cambridge, MA: MIT Press.

Veale, T. 2022. Two-fisted comics generation: Comics as a medium and as a representation for creative meanings. In *Proceedings of ICCC-22, the 13th International Conference on Computational Creativity*. 59–66.

Veale, T. 2023a. Have I Got Views For You! Generating "Fair and Balanced" Interventions into Online Debates. In *Proceedings of ICCC-23, the 14th International Conference on Computational Creativity*.

Veale, T. 2023b. The Funhouse Mirror Has Two Sides: Visual storification of debates with comics. In Campos, R.; Jorge, A.; Jatowt, A.; Bhatia, S.; and Litvak, M., eds., *Proceedings of the Text2Story'23 Workshop at ECIR, the 45th European Conference on Information Retrieval*.

Walsh, J. 2012. Comic book markup language: An introduction & rationale. *Digital humanities quart.* 6(1).

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of NeurIPS-22, the 36th Conference on Neural Information Processing Systems*.

Winters, T. 2021. Computers learning humor is no joke. *Harvard Data Science Review* 3(2).