AI Joke Generation with PlanSearch and Novelty Detection

Why did you pick the particular project?

1. The Joke Generation problem grabbed my attention immediately as I really wanted to work on this problem for a long time but never got the time to tend to it. Andrej Karpathy's legendary blog on how deep neural networks still couldn't understand humor (it was an image captioning problem back then) still resonated with many of the LLMs, where they lack quirky, unstructured and funny humor which differentiates humans from LLM-generated content. This is still a problem that resonates with me—the personality of LLMs is quite magnetic, but they lack the casual sense of language associated with truly funny content. When I started reading about computational humor, I realized most approaches were either too simplistic (template filling) or focused on joke detection rather than generation. The idea of building a system that could actually be creative, not just pattern-match, felt like a real research frontier. But what really sold me was the multi-objective optimization challenge. In most AI tasks, you optimize for one metric and call it a day. But humor? You need it to be funny (obviously), but also original (not just recycling old jokes), and diverse (not repetitive). Those goals actually conflict with each other—the funniest approach might be to copy proven joke formats, but that kills originality. That tension felt like a genuinely interesting research problem. (PS: I'm a big fan of standup comedy and trying to make people laugh, so that was also one informal and personal preference.)

If you had more compute/time, what would you have done?

2. The thing that I'm still not completely satisifed by is the human evaluation gap which differs so much as compared to the LLM as a judge which i have used to rank the jokes generated by the LLMs and displaying based on their

rankings.The key paper was "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena" by Zheng et al. (2023). They found that GPT-4 agreed with human judges about 80% of the time on pairwise comparisons, which was way higher than I expected going into this project.(so i used that as the baseline implementation along with many other novel approaches which i tried and experimented based on other papers and new ideas) but still lacks self learning and a dynamic robust system which keeps on updating. Here are the few ideas I would try if i had more compute:

- Real-time audience feedback would be the holy grail: I'd want to deploy this in actual comedy venues with audience response measurement - like, tracking laughter duration, applause intensity, even facial recognition for genuine vs. polite laughter. Then feed that back into the system so it learns what actually works in practice vs. what just sounds good on paper. The data pipeline alone would be insane.
- I'd build a proper self-improvement loop: Right now my system generates jokes, evaluates them, and that's it. But imagine if it could actually learn from its failures. I'd set up this continuous learning pipeline where the joke generator gets retrained based on which jokes consistently score well vs. poorly. Like, if wordplay jokes about tech keep bombing, it should learn to avoid that pattern. I'd probably need like 10x the compute just to run those retraining cycles.
- Advanced pruning strategies would save so much compute: Right now I'm exploring way too many bad joke branches. I'd train a separate "joke potential predictor" that could kill obviously unfunny directions early. Plus implement this hierarchical search where you first decide on comedy style, then topic, then structure, then specific words. Much more efficient than the current brute force approach.
- The multi-agent evolution thing would be incredible: I'd spawn like 50 different joke generators, each with slightly different "personalities" - one that loves puns, another that goes for absurdist humor, etc. Then I'd let them compete in this tournament-style setup where successful jokes get their generators "bred" together. Basically genetic algorithms but for comedy styles. That would require massive parallel compute but could discover entirely new humor patterns.

- The hyperparameter optimization would be a project itself: There are so many knobs to turn - temperature for generation, weights in the composite scoring, exploration vs exploitation in tree search, etc. I'd run like Bayesian optimization for weeks to find the sweet spots. Probably discover that optimal parameters change based on the target audience or comedy style.
- Multi-modal humor is where it gets really exciting: Extend beyond just text to generate memes, comic strips, even video sketches. That's like 100x more complex because now you need to coordinate visual timing with verbal timing. But imagine an AI that could generate the next viral meme format!
- The evaluation bias problem needs way more attention: I'd collect thousands of human humor ratings and really dig into where LLM judges disagree with humans. Maybe train specialized judge models that are calibrated specifically for different types of humor. Or develop this ensemble of judges with different voting strategies that includes models trained on different cultural datasets.

What did you learn in the project?

3. This project completely changed how I think about AI creativity. There are bunch of different things I learned personally which has now motivated me to pursue how to “diversify creativity in AI systems” which will not only aid in joke generation but also scientific discovery. Some of the new things that i learned both personally and for Joke Generation using AI are as follows:

- Implementing PlanSearch for creative domains was fundamentally different from traditional search problems:Unlike chess or mathematical optimization where you have clear objective functions, humor generation required adapting the UCB formula to handle subjective, multi-dimensional quality metrics. I had to modify the tree expansion strategy to generate observation paths rather than move sequences, and the state space became probabilistic rather than deterministic. The biggest insight was that creative search requires exploration of conceptually diverse branches, not just numerically optimal ones.
- Multi-objective optimization of humor dimensions was more complex than expected. My weighted ranking system combined humor (60%), diversity (30%), and novelty (10%) weights (need to refine it more with grid search for

better hyperparams).The key insight was that optimizing for pure humor often led to repetitive, clichéd jokes, while the multi-dimensional approach encouraged creative variety.

- Advanced pruning strategies were essential for making the system practical:I implemented a multi-stage pruning pipeline: quality-based observation filtering (removing low-potential comedy observations early), diversity pruning (eliminating redundant similar observations), and beam search for observation selection. The quality evaluator I built used LLM-based scoring to filter observations before expensive joke generation. This reduced computational overhead by 60% while maintaining output quality within 5% of the unpruned version.
- The evaluation suite taught me that comprehensive comparison requires standardized protocols: I built automated evaluation across multiple dimensions: humor quality (LLM-judged), novelty scoring, diversity measurement and consistency analysis The statistical significance testing revealed that small sample sizes in joke evaluation are highly unreliable -you need 20+ jokes per system for meaningful comparisons. I also learned that human evaluation baselines are essential; LLM judges can be calibrated but shouldn't be trusted without human validation.(jokes with higher ELO scores weren't necessary the funniest).

What surprised you the most?

4. I was surprised by some of the results based on the different approaches I tried which are are listed below:

- The biggest surprise was how well multi-objective optimization worked. Going in, I expected it to be a compromise like you'd get jokes that were mediocre on all dimensions rather than excellent on any one. But the 60/30/10 weighting actually produced jokes that were both funnier and more creative than single-objective approaches. It was like finding a free lunch in optimization, which supposedly doesn't exist.
- Pattern detection was both more powerful and more brittle than anticipated. My regex-based joke detection caught 89% of overused joke formats, but it also flagged genuinely creative jokes that happened to use

similar sentence structures. The semantic similarity approach using embeddings was more nuanced but computationally expensive and sometimes missed obvious paraphrases while flagging conceptually different jokes with similar vocabulary.

- Surface-Level Pattern Preference: The LLM judge consistently rated formulaic wordplay and puns higher than conceptually creative or absurdist humor, suggesting it prioritizes recognizable humor patterns over genuine creativity or surprise.
- PlanSearch Diminishing Returns: After 3-4 search iterations, joke quality plateaued or even decreased, likely due to over-optimization toward the judge's biases rather than genuine humor improvement.
- Judge Inconsistency: The same LLM judge would rate identical jokes differently when presented in different orders or contexts, even with explicit bias mitigation strategies. This inconsistency was far greater than expected (~30% variance in scores).

If you had to write a paper on the project, what else needs to be done?

5. Here are some of the key things that needs to be researched and studied so that we can write a fully fledged novel paper on Joke Generation using AI:

- Rigorous Experimental Design: The current work lacks proper experimental controls and statistical validation. I'd need to design controlled experiments with clear hypotheses, randomization protocols, and statistical power analysis. This means defining specific research questions like "Does PlanSearch generate statistically significantly funnier jokes than baseline sampling methods?" and designing experiments that can definitively answer them.
- Human Evaluation Gold Standard: The paper would require extensive human evaluation to validate LLM judge assessments. I'd need IRB approval, structured annotation guidelines, inter-annotator agreement studies, and demographically diverse evaluation panels. Without human ground truth, any claims about humor quality remain unsubstantiated.

- Baseline Comparison Studies: Currently missing systematic comparisons against established methods. The paper needs rigorous benchmarking against standard text generation approaches, existing computational humor systems (if any), and human-written jokes. Each comparison would require matched experimental conditions and statistical significance testing.
- Reproducibility Package: Academic publication requires complete reproducibility. This means standardized datasets, exact hyperparameter specifications, random seed documentation, and code that other researchers can run to replicate results. The current implementation would need extensive documentation and standardization.