Chloe Tu
ITAI 2373
October 15, 2025

L05 Computational Humor

Introduction

In this report, I am taking a deep dive into the field of computational humor. This specific area of artificial intelligence is all about trying to understand what makes us laugh and then teaching computers how to generate or recognize humor on their own. My study follows the journey of these humor models, starting from the very first simple programs that worked like basic recipes, following a strict set of rules. I then trace this path to the incredibly complex AI systems we have today, which use advanced deep learning to grasp the subtle art of comedy. To illustrate this evolution, I will carefully analyze five important models. For each one, I will explain its original purpose, the technology that makes it work, and its unique strengths and weaknesses.

After exploring these systems, I will shift my focus to a more hands-on task. I will present a manual joke-writing algorithm that I personally designed. My goal was to create a simple, step-by-step guide that anyone could use to construct a joke from scratch. I will walk through how I used this exact process to create a brand-new joke. To conclude my project, I will imagine sharing my joke with a few different people to see how they react. This final part of the report will explore how humor is not always universal and how a person's background can completely change whether they find something funny or not.

Part 1: Exploration of Computational Humor Models

Jape

My investigation started with Jape, one of the earliest and most foundational examples of a joke-generating computer program. The primary purpose of Jape was quite narrow: it was built exclusively to create punning riddles. Its mechanism was straightforward and entirely rule-based. The system relied on a large, pre-made dictionary where words were stored along with their phonetic information. Jape's program would methodically scan this dictionary to find pairs of words that sounded identical but had different meanings, known as homophones. Once it found a match, it would insert these words into a rigid joke template, such as "What do you call X?".

Jape's main strength was its role as a trailblazer. It provided crucial proof that humor, something so deeply human, could be broken down into a logical process that a machine could execute. It laid the groundwork for all future research in the field. However, its limitations were very apparent. Jape had no understanding of the words it was using; it only matched sounds. This meant its jokes often felt robotic and lacked the cleverness of human comedy. It was completely inflexible, unable to create any other type of joke or adapt its style in any way.

HahAcronym

Next, I looked into the HahAcronym system, a more specialized model designed with the sole purpose of inventing funny acronyms for any given phrase. The technology behind HahAcronym was a clever combination of templates and a large digital knowledge base called WordNet. WordNet is essentially a giant web of words connected by their meanings and relationships. When given a phrase, the system first identifies the key ideas. Then, it searches through WordNet for related, witty, or surprising words that begin with the necessary letters for the acronym. Finally, it arranges these words into a humorous, template-based sentence.

The greatest strength of HahAcronym is its ability to use these semantic connections to create genuinely clever and relevant jokes. The acronyms it generates often feel surprisingly insightful. However, its primary limitation is its highly specialized nature. It is a one-trick pony, capable only of producing this single type of humor. The funniness of its output is also completely dependent on the quality of its pre-written templates and the richness of the WordNet database. If a concept is not well-represented in its knowledge base, it cannot generate a good acronym for it.

STANDUP

The STANDUP system was particularly interesting to me because its purpose goes beyond mere entertainment. It was designed as an interactive tool to assist children with communication disabilities, using humor as a way to make learning fun. The system generates simple punning riddles to help these children practice social interaction, language comprehension, and turn-taking in conversation. Its mechanism works by identifying words with sound-based ambiguities and then building an interactive riddle around them. The child is then prompted to solve the riddle, encouraging them to think about language in a playful way.

The most significant strength of STANDUP is this brilliant and compassionate application of computational humor. It transforms a joke generator into a meaningful therapeutic and educational tool, demonstrating that AI humor can have a positive impact on people's lives. The system's main limitation is that its humor is intentionally simple and tailored for a young audience. The riddles and puns it creates are not designed to be funny to adults and are confined to that specific format, making it less versatile as a general humor generator.

BERT-Based Humor Detection

After looking at joke generators, I turned my attention to a different challenge: humor detection. For this, I studied systems built using BERT, a powerful and modern AI language model. The purpose of a BERT-based system is not to create jokes, but to read and analyze a piece of text and accurately determine if it is humorous. The mechanism behind it is complex; BERT is pre-trained by reading a massive portion of the internet, allowing it to learn the nuances, context, and patterns of human language. It reads a sentence bidirectionally, meaning it considers the context from both left and right at the same time, giving it a much deeper understanding than older models.

Its primary strength is its remarkable accuracy. BERT can often detect sophisticated forms of humor like sarcasm or subtle wordplay that would have been impossible for earlier systems to recognize. This makes it useful for tasks like content moderation or analyzing audience sentiment. The main limitation, however, is that it needs a huge, carefully labeled dataset to be fine-tuned for the specific task of humor detection. It can also be slow to adapt to new forms of internet slang or memes, as its knowledge is based on the data it was trained on.

The HumorHawk System

Finally, I explored the HumorHawk system, which represents the current state-of-the-art in computational humor. HumorHawk is designed for both the generation of original humorous content and the assessment of its quality. Its core mechanism is based on deep learning, using advanced neural networks like those that power models like GPT. A neural network functions like a simplified version of the human brain, with interconnected nodes that learn patterns from data. HumorHawk is trained on an enormous dataset of human jokes, stories, and scripts. By analyzing this data, it learns the underlying structures and rhythms of comedy.

The greatest strength of HumorHawk is its amazing flexibility and creativity. Because it is not limited by rigid templates, it can generate a wide variety of humor, from one-liners to short, funny stories, and can even attempt to mimic specific comedic styles. Its most significant weakness is its lack of real-world understanding. While it is excellent at mimicking patterns, it does not truly comprehend the meaning behind the jokes. This can lead to it producing content that is grammatically correct and structurally sound, but logically nonsensical or absurd in a way that is not funny.

Part 2: Design of a Manual Joke Algorithm

I decided to create my own manual algorithm for joke construction to better understand the systematic process behind humor. My goal was to design a simple, four-step method that could guide the creation of a pun or a basic one-liner. I focused on making the steps logical and easy for anyone to follow.

The Algorithm

1. **Establish a Core Concept:** The first and most important step is to choose a familiar subject. This provides a shared context between the joke teller and the audience. When a topic is relatable, like school, technology, or animals, the audience already has a set of expectations, which is crucial for the misdirection that comes later.
2. **Identify an Ambiguity:** Within that core concept, the next step is to find a word or phrase that has a double meaning. This ambiguity is the engine of the joke. It can be a homophone (words that sound the same but have different meanings) or a word with multiple distinct definitions. This element is what allows for the surprise twist.
3. **Create Misdirection:** With the ambiguous word in mind, I then construct the setup of the joke. This sentence or question is carefully worded to lead the audience's mind toward the most obvious or expected meaning of the ambiguous word. This is a psychological trick to build a specific expectation.
4. **Formulate the Punchline:** The final step is to deliver the punchline. The punchline sharply pivots to the second, unexpected meaning of the ambiguous word. This sudden violation of the audience's expectation creates a cognitive jolt, and that moment of surprise and resolution is what triggers laughter.

Sample Joke Generation

I put my four-step algorithm into practice to create a sample joke.

1. **Core Concept:** I chose the topic of computer programming. I felt this was a good choice because it is filled with technical jargon that sounds like normal English words, making it a perfect environment for puns.
2. **Identify an Ambiguity:** I brainstormed a few terms and landed on the word "arrays." In the world of programming, an array is a fundamental data structure used to store a collection of elements. However, phonetically, it sounds exactly like the common workplace phrase "a raise," meaning a salary increase.
3. **Create Misdirection:** To build the setup, I made up a question that would make the audience think about typical job-related issues. My question was: "Why did the

programmer quit his job?" This immediately makes the listener anticipate answers related to stress, a bad boss, or money.

4. **Formulate the Punchline:** For the final step, I delivered the punchline that reveals the second meaning: "Because he didn't get arrays." The humor comes from the abrupt switch from an expected human resources issue to a technical programming one. The misdirection is resolved in a surprising and silly way.

Part 3: Human Evaluation

As a full-time AI college student, my academic interests often lead to fascinating discussions with friends from diverse backgrounds. To perform a human evaluation of the joke I generated, I presented it to a close friend of mine who is from Saudi Arabia. She is fluent in English, but it is not her native language. I was interested to see how humor would translate across both a linguistic and cultural divide. Her reaction provided a powerful and insightful lesson on the complexities of humor perception.

When I told her the joke, her initial reaction was one of polite confusion. She understood every word in the setup and the punchline, but the humorous connection was entirely absent. I explained that "arrays" sounds identical to "a raise," and after a moment, she acknowledged the linguistic trick. However, she did not laugh or find it funny. Her feedback allowed me to analyze the joke's failure on two distinct levels. The first was the obvious linguistic barrier. Puns are a notoriously language-specific form of humor that relies on accidental phonological similarities. This sound-based connection, which is second nature to a native English speaker, is not intuitive and can be easily missed by someone who learned the language formally.

The second, more subtle barrier was cultural. She explained that while she understood the concept, the style of humor felt very foreign. In her experience, humor is often more situational, narrative-driven, or based on social observation rather than on clever linguistic manipulation. The entire premise of a pun felt more like a riddle or a word puzzle than a genuine joke. This single piece of feedback was incredibly illuminating. It demonstrated clearly that humor is not a universal constant; it is deeply embedded in language and cultural context. This experience underscored the immense challenge facing AI: to create a truly sophisticated computational humor system, it must be able to model not just language, but the vast and varied cultural frameworks in which that language exists.

Reference

- **For Jape**
  - Binsted, Kim. Machine Humour: An Implemented Model of Puns. 1996. https://www2.hawaii.edu/~binsted/papers/Binstedthesis.pdf

- **For HahAcronym**
  - Stock, Oliviero, and Carlo Strapparava. HAHAcronym: A Computational Humor System. 2005. https://aclanthology.org/P05-3029.pdf

- **For STANDUP**
  - Ritchie, Graeme, et al. A Practical Application of Computational Humour. https://homepages.abdn.ac.uk/g.ritchie/pages/papers/ijwcc07.pdf

- **BERT-based Humor Detection**
  - Weller, Orion, and Kevin Seppi. Humor Detection: A Transformer Gets the Last Laugh. 2019. https://aclanthology.org/D19-1372.pdf

- **The HumorHawk System**
  - Donahue, David, et al. "HumorHawk at SemEval-2017 Task 6: Mixing Meaning and Sound for Humor Recognition." Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022), 1 Jan. 2017, pp. 98–102, aclanthology.org/S17-2010/, https://doi.org/10.18653/v1/s17-2010.