Yankovinator

Song Parody Generator using NaturalLanguage and Ollama

Shyamal Suhana Chandra

November 22, 2025

Overview

- **What**: Convert songs into parodies with theme constraints
- **How**: NaturalLanguage framework + Ollama LLM
- **Why**: Maintain syllable structure while generating creative parodies
- **Language**: Swift 6.2+

Architecture

Core Components:

- SyllableCounter
- OllamaClient
- ParodyGenerator
- Yankovinator API

Technology:

- NaturalLanguage
- Ollama API
- AsyncHTTPClient
- ArgumentParser

How It Works

1. Analyze original song's syllable structure
2. Detect rhyming scheme automatically
3. Extract theme keywords and definitions
4. Generate each line with:
   - Exact syllable matching (word-by-word)
   - Rhyming constraints
   - Semantic coherence with previous lines
   - Theme advancement
5. Refine for semantic coherence
6. Apply capitalization and punctuation matching
7. Maintain rhythm, style, and meaning

```
let parody = try await Yankovinator.generateParody(
    originalLyrics: lyrics,
    keywords: keywords
)
```

Syllable Counting

- Uses NaturalLanguage tokenization
- Vowel counting heuristics
- Special case handling:
  - Silent 'e' endings
  - 'le' endings
  - Consonant clusters
- Per-line and per-word analysis
- Proper handling of contractions (don't, can't, it's)

Example Usage

Input:

```
Twinkle twinkle little star
How I wonder what you are
```

Keywords:

```
space: the physical universe
stars: luminous celestial bodies
```

Output: Parody with matching syllables and theme

Features

- ✓Syllable-accurate parody generation
- ✓Word-by-word syllable matching
- ✓Automatic rhyme detection and enforcement
- ✓Semantic coherence across lines
- ✓Theme advancement (not just mention)
- ✓Capitalization and punctuation matching
- ✓NaturalLanguage framework integration
- ✓Ollama LLM support (llama3.2:3b)
- ✓Command-line interface
- ✓Comprehensive testing
- ✓Proper grammar with contractions

CLI Usage

```
yankovinator lyrics.txt \
  --keywords themes.txt \
  --ollama-url http://localhost:11434 \
  --model llama3.2 \
  --output parody.txt \
  --analyze \
  --verbose
```

Testing

- Unit tests for syllable counting
- Integration tests for parody generation
- Keyword extraction tests
- XCTest framework

Run: `swift test`

Requirements

- Swift 6.3+
- macOS 13+ or iOS 16+
- Ollama installed and running
- Model: llama3.2 (or compatible)

Installation

Homebrew (Recommended):

1. `brew tap shyamalschandra/yankovinator`
2. `brew install yankovinator`
3. Start Ollama: `ollama serve`
4. Pull model: `ollama pull llama3.2:3b`
5. Run: `yankovinator lyrics.txt`

From Source:

1. Clone repository
2. `swift build`
3. `swift run yankovinator`

API Reference

- `Yankovinator.generateParody()` - Main entry point
- `SyllableCounter.countSyllables()` - Word analysis
- `SyllableCounter.analyzeSongStructure()` - Song analysis
- `ParodyGenerator.generateParody()` - Full generation
- `RhymeSchemeAnalyzer.detectRhymeScheme()` - Rhyme detection
- `OllamaClient.generateKeywords()` - Keyword generation

Reference Manual: Complete API documentation available in `reference.pdf`

Algorithm Flow

1. Parse input lyrics
2. Analyze syllable structure (total and word-by-word)
3. Detect rhyming scheme
4. Load theme keywords
5. For each line:
   - Build prompt with constraints and context
   - Request from Ollama with previous lines
   - Validate syllable count
   - Refine word-by-word matching
   - Refine semantic coherence
   - Apply capitalization and punctuation
   - Clean and format
6. Return complete parody

Benefits

- Maintains original rhythm
- Creative theme integration
- Accurate syllable matching (word-by-word)
- Semantic coherence across lines
- Automatic rhyme detection
- Theme advancement (not just mention)
- Easy to use API
- Extensible architecture

Semantic Coherence

- Context-aware generation (8 previous lines)
- Theme advancement (not just mention)
- Narrative flow maintenance
- Consistent imagery and metaphors
- Logical progression
- Dedicated semantic refinement pass

Capitalization & Punctuation

- Extracts capitalization pattern from original
- Preserves exact punctuation placement
- Word-by-word capitalization matching
- Maintains spacing and structure
- Automatic pattern application
- Ensures style consistency

Future Enhancements

- Multiple model support
- Real-time generation
- Batch processing
- Web interface
- Advanced theme analysis
- Multi-language support

Documentation

- **Technical Paper**: Architecture and algorithms
- **Presentation**: This slide deck
- **Reference Manual**: Complete API documentation
- All available on GitHub Pages

Questions?

Thank you!

`github.com/shyamalschandra/Yankovinator`

Documentation: `github.com/shyamalschandra/Yankovinator`