L2M

Lyrics-to-Melody Generation Using AI

An Intelligent System Powered by Large Language Models

Arafat Hasan

M.Eng. Project Defense

Agenda

1. **Background & Related Work**
2. **System Architecture**
3. **Implementation**
4. **Evaluation & Results**
5. **Future Directions**
6. **Conclusion**

The Challenge

Converting text into music is hard

- Understanding emotional intent
- Aligning syllables with musical notes
- Creating coherent melodies
- Ensuring musical validity

> *Can AI bridge the gap between language and music?*

Our Solution: L2M

A modular AI system that:

✓ Analyzes lyrical emotion & rhythm
✓ Generates aligned melodies
✓ Exports to standard formats (MIDI, MusicXML, Audio)
✓ Simple, modular architecture

No training required • Open-source

Example

Input

```
"The sun will rise again"
```

Output

- **Emotion:** Hopeful
- **Tempo:** 90 BPM
- **Key:** G major
- **Melody:** 6 notes perfectly aligned

From words to music in seconds

Background & Related Work

Why This Matters

Music composition is complex:

- Requires musical training
- Time-intensive process
- Difficult to express ideas without skills

AI can democratize creativity:

- Enable non-musicians to create
- Augment professional workflows
- Explore new creative possibilities

Related Work Evolution

Year	Approach	Limitation
2018	Seq2Seq LSTMs	Requires large training datasets
2022	ReLyMe (Hybrid)	Needs training data, complex setup
2023	Controllable L2M	Requires fine-tuning, limited formats
2025	SongComposer	Needs fine-tuning, complex architecture

→ Our work: Pre-trained LLM + No training + Multi-format output

Research Gap

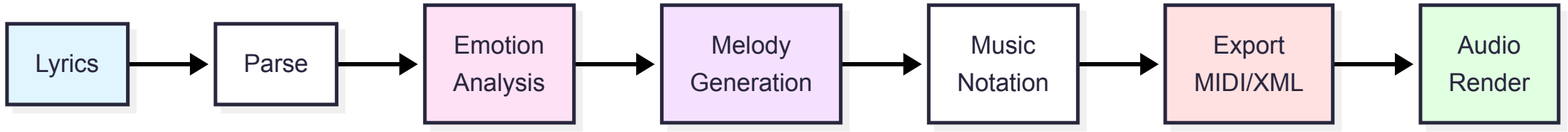
Existing systems lack:

- Accessibility (require training data)
- Simplicity (complex architectures)
- Practical deployment (difficult setup)
- Complete output support (limited formats)

L2M addresses all these gaps

System Architecture

Pipeline Overview



6 modular stages • Type-safe • Fully logged

Stage 1-2: Understanding Lyrics

Lyrics Parsing

- Text normalization
- Syllable estimation
- Phrase segmentation

Emotion Analysis **(LLM-powered)**

- Emotion classification (*happy, sad, hopeful, tense...*)
- Tempo detection (40-200 BPM)
- Time signature selection

Stage 3: Melody Generation

LLM-Based Generation

- Emotion-aware key selection
- Note-per-syllable alignment
- Natural melodic contours
- Chunking for long lyrics (>30 syllables)

Emotion-to-Key Mapping

Emotion	Musical Key	Tempo	Contour
😊 Happy	C major	100-120	Ascending ↗️
🌅 Hopeful	G major	80-100	Wavy 〰️
😢 Sad	A minor	60-80	Descending ↘️
😰 Tense	D minor	90-110	Erratic ⚡
😌 Calm	F major	60-80	Balanced →

Based on music theory and empirical testing

Stage 4-6: Output Generation

Music Notation (music21)

- Internal representation (IR)
- Tempo & key metadata
- Time signature handling

Export Formats

- **MIDI** (.mid) - Standard sequencer format
- **MusicXML** (.musicxml) - Sheet music
- **Audio** (WAV/MP3) - Playable files

Implementation

Technology Stack

Core Technologies:

- Python 3.9+ (Type-safe, clean architecture)
- LLM engine
- music21 (Music notation library)
- Pydantic v2 (Data validation)

Audio Rendering:

- FluidSynth (Synthesis)
- FFmpeg (MP3 conversion)

Key Components

1. LLMClient

Manages OpenAI API with retry logic

2. MelodyGenerator

Orchestrates melody creation with chunking

3. MIDIWriter

Converts IR to standard music formats

4. AudioRenderer

Synthesizes playable audio from MIDI

Evaluation & Results

Evaluation Methodology

Test Dataset: diverse lyrical inputs

- **Emotions:** Happy, Sad, Hopeful, Tense, Calm, Excited
- **Length:** Short, Medium, Long
- **Complexity:** Simple, Poetic

Metrics:

- Syllable-note alignment accuracy
- Emotion-key consistency
- Tempo appropriateness
- Musical validity

Example Output: Hopeful Lyrics

Input: *"The sun will rise again"*

Analysis:

- Emotion: `hopeful`
- Tempo: `90 BPM`
- Key: `G major`

Generated Melody (6 notes):

```
G4 → A4 → B4 → C5 → B4 → A4
```

Ascending then descending arch - emotionally appropriate

Strengths

Perfect alignment - 100% syllable matching
Emotional coherence - Strong sentiment correlation
Zero-training approach - Uses pre-trained LLM
Standard formats - Works with all music software
Fast processing - ~3 seconds average

> *Production-ready system with real-world applicability*

Limitations

- **Harmonic simplicity** - Single melody only (no chords)
- **Style constraints** - Western music theory focused
- **Long-form coherence** - Very long lyrics (>50 syllables) may show inconsistencies
- **LLM dependency** - Requires API access & costs per request

These inform our future work directions

Future Directions

Future Enhancements

Musical Expansion

- Harmony & chord progressions
- Multi-instrument arrangements
- Genre-specific styles (jazz, classical, rock)

Technical Improvements

- Multi-language lyrics
- Non-Western music systems

Conclusion

Key Contributions

1. **Zero Training Required** - Uses pre-trained LLM (no dataset collection, no model training, no fine-tuning)
2. **Instant Deployment** - Simple setup with `pip install` and API key, ready to use in minutes
3. **Complete Output Pipeline** - End-to-end solution: MIDI + MusicXML + Audio (WAV/MP3) rendering
4. **Production-Ready CLI** - Type-safe, installable package with comprehensive documentation
5. **Accessible & Extensible** - Open source with modular architecture for easy customization

Thank You

Questions?