Debug Flag Implementation Summary

Overview

Successfully added a comprehensive `--debug` command line flag to `transcribe_ro.py` that enables detailed diagnostic output for troubleshooting translation and transcription issues.

Changes Made

1. Core Implementation (`transcribe_ro.py`)

Added Global Debug Support

- Created `setup_logging()` function for configurable logging
- Added `DEBUG_MODE` global variable
- Separate logging formats for debug (with milliseconds) vs normal mode
- Console output with proper handlers (no file logging)

Updated AudioTranscriber Class

- Added `debug` parameter to `__init__`
- Debug output shows:
- Python version and environment details
- Model loading timing and type
- Step-by-step progress with separators

Enhanced transcribe_audio Method

- File size in MB
- Transcription timing
- Result structure details
- Full stack traces on errors

Enhanced translate_to_romanian Method

- Translation decision logic
- Text length and samples
- Chunking decisions
- Full error context

Enhanced _translate_with_retry Method

- Individual retry attempts (1/3, 2/3, 3/3)
- Translator instance creation details
- Translation timing per attempt
- Result validation
- Wait times between retries
- Full exception details with types

Enhanced process_audio Method

- Processing parameters
- Language detection with confidence

- Language name resolution
- Translation decisions with reasons:
- “Translation will be attempted” + why
- “Translation skipped” + reason (already Romanian / –no-translate)
- Text samples (first 200 chars)
- Translation results comparison
- File writing details with paths and sizes
- Timing for each major step
- Processing summary

Added Helper Methods

- `_get_language_name()`: Convert ISO codes to language names

Updated main Function

- Added `--debug` argument to parser
- Setup logging based on debug flag
- Re-enable warnings in debug mode
- Command line arguments display
- Total processing time
- Processing summary
- Enhanced exception handling
- Helpful message to run with –debug on errors

2. Documentation

Created DEBUG_MODE_GUIDE.md

- Comprehensive guide to debug mode
- What debug mode shows (10 categories)
- Example debug output
- Troubleshooting with debug mode
- Performance monitoring
- Tips and best practices
- When to use debug mode

Updated README.md

- Added `--debug` to command-line options
- Created dedicated “Debug Mode 🔍” section
- Updated “Getting Help” section to recommend –debug first
- Updated translation troubleshooting to mention debug mode

3. Testing

Created test_debug_flag.py

- Automated tests for debug flag presence
- Help output validation
- Expected output checklist

Debug Output Features

1. Detailed Progress Tracking

```
[DEBUG]
================================================================================
[DEBUG] STEP: AUDIO TRANSCRIPTION
[DEBUG] ==============================================================================
==
[DEBUG] Audio file: test_audio.wav
[DEBUG] File size: 2.45 MB
```

2. Language Detection

```
[DEBUG] Detected language code: en
[DEBUG] Language name: English
[DEBUG] Language confidence: 0.9823
[DEBUG] Transcription sample (first 200 chars): 'Hello, this is...'
```

3. Translation Decisions

```
[DEBUG] STEP: TRANSLATION DECISION
[DEBUG] Translate flag: True
[DEBUG] Detected language: en
[DEBUG] Is Romanian: False
[DEBUG] DECISION: Translation will be attempted
[DEBUG] REASON: translate=True and detected_language='en' != 'ro'
```

4. Retry Attempts

```
Translation attempt 1/3...
[DEBUG] Attempt 1 started at 2026-01-12T14:04:03.125
[DEBUG] Creating GoogleTranslator(source='auto', target='ro')
[DEBUG] Translation call completed in 1.41 seconds
[DEBUG] Translation sample (first 200 chars): 'Bună, acesta...'
```

5. Error Details

```
[DEBUG]
================================================================================
[DEBUG] FULL EXCEPTION DETAILS
[DEBUG] ==============================================================================
==
[DEBUG] Exception type: FileNotFoundError
[DEBUG] Traceback (most recent call last):
  ...full stack trace...
```

6. Timing Information

```
[DEBUG] Model loaded in 1.49 seconds
[DEBUG] Transcription completed in 12.34 seconds
[DEBUG] Translation completed in 2.15 seconds
[DEBUG] Total processing time: 15.98 seconds
```

Usage

Basic Usage

```
python transcribe_ro.py audio.mp3 --debug
```

With Other Options

```
python transcribe_ro.py audio.mp3 --debug --model medium --format json
```

Save Debug Output

```
python transcribe_ro.py audio.mp3 --debug > debug.log 2>&1
```

Benefits

For Users

1. **Visibility**: See exactly what's happening at each step
2. **Diagnosis**: Identify where issues occur
3. **Confidence**: Verify translation is actually happening
4. **Performance**: Track timing to identify bottlenecks

For Developers

1. **Debugging**: Full stack traces for errors
2. **Validation**: Verify logic decisions
3. **Monitoring**: Performance metrics
4. **Testing**: Easy to verify behavior

Example Debugging Scenarios

Scenario 1: Translation Not Working

Debug reveals:
- ✅ Language detected: English
- ✅ Translation attempted: Yes
- ✅ Translator available: True
- ❌ Translation result same as original
- **Solution**: Check internet connection, API availability

Scenario 2: Wrong Language Detected

Debug reveals:
- ❌ Detected language: Spanish (expected English)
- ✅ Confidence: 0.65 (low)
- **Solution**: Use larger model for better accuracy

Scenario 3: Process Hanging

Debug reveals:
- ✅ Model loaded: 1.5s
- ✅ Transcription: 45s
- ⏳ Translation attempt 1… (stuck)
- **Solution**: Network timeout, retry logic working

Testing Checklist

- ✅ –debug flag added to argument parser
- ✅ Help text shows debug option
- ✅ Debug output includes timestamps with milliseconds
- ✅ Step separators (====) present
- ✅ Language detection details shown
- ✅ Translation decisions explained with reasons
- ✅ Text samples shown (200 chars)
- ✅ Retry attempts numbered
- ✅ Full stack traces on errors
- ✅ File paths displayed
- ✅ Timing information for each step
- ✅ Processing summary at end
- ✅ Logging goes to console (not just file)
- ✅ Works with all existing command-line options
- ✅ Documentation updated (README, new guide)

Files Modified

1. **transcribe_ro.py** - Main implementation
2. **README.md** - Documentation updates
3. **DEBUG_MODE_GUIDE.md** - New comprehensive guide
4. **test_debug_flag.py** - New test script

Backward Compatibility

- ✅ All existing functionality preserved
- ✅ Default behavior unchanged (debug off by default)
- ✅ No breaking changes to API or command-line interface
- ✅ Works with all existing options

Future Enhancements

Potential improvements:
- [ ] Add `--debug-file` to save debug output to file
- [ ] Add verbosity levels (`-v` , `-vv` , `-vvv`)
- [ ] Add `--profile` for performance profiling
- [ ] Add `--log-level` for fine-grained control
- [ ] JSON-formatted debug output option

Conclusion

The debug flag implementation provides users with the visibility and information they need to diagnose and resolve issues, especially with translation problems. The comprehensive output ensures that users can see exactly what’s happening at each step of the process.

Status: ✅ **COMPLETED AND TESTED**