Models Documentation - Requirements Extractor

Version: 1.0
Date: November 2024

Table of Contents

Overview

The Requirements Extractor uses a two-stage AI pipeline:

1. **Stage 1: Transcription** - Converts audio/video to text
    - **Local Option**: OpenAI Whisper (openai-whisper library)
    - **Cloud Option**: OpenAI Whisper API (whisper-1 model)
2. **Stage 2: Requirements Extraction** - Extracts structured requirements from text
    - **Local Option**: Ollama (llama3.2, mistral, or other local LLMs)
    - **Cloud Option**: OpenAI GPT models (gpt-4o-mini, gpt-4o, gpt-3.5-turbo)

Transcription Models (Audio to Text)

1. Local Whisper (openai-whisper)

Library: openai-whisper
Model Type: Automatic Speech Recognition (ASR)
Default Model Size: base

Model Variants Available

Model Size	Parameters	VRAM Required	Speed	Accuracy	Disk Space
tiny	39M	~1 GB	Fastest	Good	~75 MB
base	74M	~1 GB	Fast	Better	~142 MB
small	244M	~2 GB	Medium	Very Good	~466 MB
medium	769M	~5 GB	Slow	Excellent	~1.5 GB
large	1550M	~10 GB	Slowest	Best	~3 GB
large-v2	1550M	~10 GB	Slowest	Best	~3 GB

large-v3	1550M	~10 GB	Slowest	Best	~3 GB

Technical Details

- **Architecture**: Transformer-based encoder-decoder
- **Training Data**: 680,000 hours of multilingual and multitask supervised data
- **Languages Supported**: 99+ languages
- **Audio Formats**: MP3, WAV, M4A, FLAC, and more (via FFmpeg)
- **Processing**: Runs entirely on local machine
- **API Key Required**: No

Current Implementation

```
# Current default: base model
model = whisper.load_model("base")
result = model.transcribe(audio_path)
text = result.get("text", "")
```

Code Location: app.py, function transcribe_audio_local_whisper()

Performance Characteristics

- **Processing Speed**: ~1x real-time (base model on CPU)
- **GPU Acceleration**: Significantly faster with CUDA/ROCm
- **Memory Usage**: Model size + audio buffer
- **Accuracy**: ~95%+ word accuracy for clear audio

Advantages

No API key required
No data sent to external servers (privacy)
No usage costs
Works offline
Supports 99+ languages

Disadvantages

Slower than cloud API
Requires significant disk space for larger models
Requires FFmpeg for audio processing
Higher memory usage for large models

---

2. OpenAI Whisper API (whisper-1)

Service: OpenAI API
Model Name: whisper-1
Model Type: Automatic Speech Recognition (ASR)
Endpoint: https://api.openai.com/v1/audio/transcriptions

Technical Details

- **Architecture**: Same as local Whisper (large-v2 equivalent)

- **Processing**: Cloud-based, runs on OpenAI servers
- **File Size Limit**: 25 MB per request
- **Audio Formats**: MP3, MP4, MPEG, MPGA, M4A, WAV, WEBM
- **API Key Required**: Yes

Current Implementation

```
client = OpenAI(api_key=api_key)
transcript = client.audio.transcriptions.create(
    model="whisper-1",
    file=audio_file,
    response_format="text"
)
text = transcript.text
```

Code Location: app.py, function transcribe_audio_with_whisper()

Chunking for Large Files

For files > 25 MB, the system automatically:

1. Splits audio into ~20-minute chunks using pydub
2. Transcribes each chunk separately
3. Combines transcripts in order

Code Location: app.py, lines 556-671

Performance Characteristics

- **Processing Speed**: ~0.1x real-time (much faster than local)
- **Latency**: Network-dependent (typically 2-10 seconds)
- **Accuracy**: ~98%+ word accuracy
- **Cost**: ~$0.006 per minute of audio

Advantages

Very fast processing
High accuracy
No local resources required
Automatic chunking for large files
No FFmpeg setup needed

Disadvantages

Requires API key
Data sent to OpenAI servers
Usage costs
Requires internet connection
25 MB file size limit per request

Requirements Extraction Models (Text to Structured Requirements)

1. OpenAI GPT Models

Service: OpenAI API
Model Type: Large Language Model (LLM)
Endpoint: https://api.openai.com/v1/chat/completions

Available Models

Model	Parameters	Context Window	Speed	Cost (per 1K tokens)	Best For
gpt-4o-mini	~7B	128K	Fast	$0.15/$0.60	Default, cost-effective
gpt-4o	~1.7T	128K	Medium	$2.50/$10.00	Higher accuracy needed
gpt-3.5-turbo	~175B	16K	Fastest	$0.50/$1.50	Legacy, lower cost

Default Model: gpt-4o-mini

Technical Details

- **Architecture**: Transformer-based decoder
- **Training**: Pre-trained on large text corpus, fine-tuned for chat
- **Response Format**: JSON object (enforced via response_format)
- **Temperature**: 0.3 (for consistent, deterministic output)
- **System Prompt**: Expert business analyst persona

Current Implementation

```
response = client.chat.completions.create(
    model=model,  # e.g., "gpt-4o-mini"
    messages=[
        {
            "role": "system",
            "content": "You are an expert business analyst..."
        },
        {
            "role": "user",
            "content": prompt
        }
    ],
    response_format={"type": "json_object"},
    temperature=0.3
)
result = json.loads(response.choices[0].message.content)
```

Code Location: requirements_extractor.py, lines 266-283

Prompt Structure

The system uses a structured prompt that includes:

1. **System Message**: Defines the AI's role as a business analyst
2. **User Prompt**: Contains:
    - Full conversation transcript

    - Extraction instructions
    - JSON schema specification
    - Examples of expected output

Code Location: requirements_extractor.py, function _create_extraction_prompt()

Chunking Strategy

For large transcripts (>50 messages):

- Splits into chunks of 50 messages
- Processes each chunk independently
- Merges results and deduplicates
- Removes duplicate requirements based on description similarity

Code Location: app.py, function extract_requirements(), lines 814-870

Performance Characteristics

- **Processing Speed**:
    - gpt-4o-mini: ~2-5 seconds per chunk
    - gpt-4o: ~5-15 seconds per chunk
    - gpt-3.5-turbo: ~1-3 seconds per chunk
- **Token Usage**: ~500-2000 tokens per chunk (input + output)
- **Accuracy**: High for structured extraction tasks

Advantages

High accuracy for structured extraction
Fast processing
Reliable JSON output
Good at following instructions
Handles complex requirements well

Disadvantages

Requires API key
Usage costs
Data sent to OpenAI servers
Requires internet connection

---

2. Ollama (Local LLMs)

Service: Local Ollama server
Model Type: Large Language Model (LLM)
Endpoint: http://localhost:11434/api/generate

Available Models

Model	Parameters	Context Window	VRAM Required	Best For
llama3.2	3B	128K	~4 GB	Default, balanced

llama3.2:3b	3B	128K	~4 GB	Smaller variant
mistral	7B	8K	~6 GB	Alternative option
codellama	7B-34B	16K-100K	6-20 GB	Code-focused
llama3	8B-70B	8K-128K	8-40 GB	Larger options

Default Model: llama3.2

Technical Details

- **Architecture**: Various (depends on model)
- **Processing**: Runs entirely on local machine
- **API**: OpenAI-compatible API via Ollama server
- **Temperature**: 0.3 (for consistency)
- **Timeout**: 300 seconds (5 minutes) per request

Current Implementation

```
response = requests.post(
    "http://localhost:11434/api/generate",
    json={
        "model": ollama_model,  # e.g., "llama3.2"
        "prompt": full_prompt,
        "stream": False,
        "options": {
            "temperature": 0.3
        }
    },
    timeout=300
)
result_text = response.json().get("response", "")
# Extract JSON from response (may have extra text)
json_match = re.search(r'\{.*\}', result_text, re.DOTALL)
result = json.loads(json_match.group())
```

Code Location: requirements_extractor.py, lines 263-283

Health Check

The system checks if Ollama is running before use:

```
response = requests.get("http://localhost:11434/api/tags", timeout=3)
if response.status_code == 200:
    # Ollama is available
    models = response.json().get('models', [])
```

Code Location: requirements_extractor.py, lines 184-199

Performance Characteristics

- **Processing Speed**:
    - llama3.2 (3B): ~5-20 seconds per chunk (CPU)

  - llama3.2 (3B): ~2-5 seconds per chunk (GPU)
  - Larger models: Slower but more accurate

- **Memory Usage**: Model size + context buffer
- **Accuracy**: Good for structured tasks, may need prompt tuning

Advantages

No API key required
No data sent externally (privacy)
No usage costs
Works offline
Full control over model selection

Disadvantages

Requires local installation and setup
Slower than cloud API (especially on CPU)
Requires significant RAM/VRAM
May need prompt engineering for best results
JSON extraction may need regex parsing (less reliable)

Text Parsing (Post-Transcription)

After transcription, the raw text is parsed into structured message format before requirements extraction.

Parsing Methods

1. Simple Text Parsing (for Whisper output)

Whisper outputs plain text without speaker identification. The system:

```
lines = transcript_text.split('\n')
messages = []
for line in lines:
    line = line.strip()
    if line:
        messages.append({
            'speaker': 'Speaker',  # Generic speaker
            'text': line,
            'timestamp': None
        })
```

Code Location: app.py, lines 752-761

2. Structured Transcript Parsing

For pre-formatted transcripts (TXT, VTT, JSON), the TranscriptParser class extracts:

- **Speaker names**: From patterns like "Speaker: text" or "[Speaker] text"
- **Timestamps**: From VTT format or JSON structure
- **Message text**: Cleaned and formatted

Code Location: requirements_extractor.py, class TranscriptParser

Parsing Patterns

Text Format:

```
John Doe: We need user authentication
Jane Smith: That's a good point
```

VTT Format:

```
WEBVTT

00:00:10.000 --> 00:00:15.000
John Doe: We need user authentication

00:00:15.000 --> 00:00:20.000
Jane Smith: That's a good point
```

JSON Format:

```
{
  "messages": [
    {
      "speaker": "John Doe",
      "text": "We need user authentication",
      "timestamp": "00:00:10"
    }
  ]
}
```

Model Comparison

Transcription Models

Feature	Local Whisper (base)	OpenAI Whisper API
Speed	~1x real-time	~0.1x real-time
Accuracy	~95%	~98%
Cost	Free	~$0.006/min
Privacy	Local only	Cloud
Setup	Requires FFmpeg	Just API key
File Size	Unlimited	25 MB limit
Languages	99+	99+
Offline	Yes	No

Requirements Extraction Models

Feature	OpenAI GPT (gpt-4o-mini)	Ollama (llama3.2)
Speed	~2-5 sec/chunk	~5-20 sec/chunk (CPU)
Accuracy	High	Good

Cost	~$0.15/1K tokens	Free
Privacy	Cloud	Local
Setup	Just API key	Install + setup
JSON Reliability	Excellent	⚠ May need parsing
Context Window	128K	128K
Offline	No	Yes

Model Selection Guide

When to Use Local Whisper

You want complete privacy
You process many files (cost savings)
You have sufficient disk space
You don't mind slower processing
You work offline frequently

When to Use OpenAI Whisper API

You need fast processing
You want highest accuracy
You have API budget
You process occasional files
You want minimal setup

When to Use OpenAI GPT Models

You need highest extraction accuracy
You want reliable JSON output
You have API budget
You process many different requirement types
You want consistent results

When to Use Ollama

You want complete privacy
You process many files (cost savings)
You have sufficient RAM/VRAM
You don't mind slower processing
You want full control

Recommended Combinations

Privacy-First Setup

- Local Whisper + Ollama
- Complete privacy
- No costs
- ⚠ Slower processing

Balanced Setup

- Local Whisper + OpenAI GPT
- Privacy for transcription
- Fast, accurate extraction
- ⚠ Extraction costs

Speed-First Setup

- OpenAI Whisper API + OpenAI GPT
- Fastest overall
- Highest accuracy
- All data to cloud
- Highest costs

Cost-Effective Setup

- Local Whisper + Ollama
- No costs
- Privacy
- ⚠ Requires local resources

Technical Specifications

Model Input/Output Formats

Whisper Input

- **Format**: Audio file (MP3, WAV, M4A, etc.)
- **Size**: Unlimited (local), 25 MB (API)
- **Sample Rate**: Auto-detected
- **Channels**: Mono or stereo

Whisper Output

- **Format**: Plain text string
- **Language**: Detected automatically
- **Punctuation**: Included
- **Speaker IDs**: Not included (single speaker assumed)

GPT/Ollama Input

- **Format**: Text string (conversation transcript)
- **Structure**: Formatted as conversation
- **Length**: Up to context window limit
- **Encoding**: UTF-8

GPT/Ollama Output

- **Format**: JSON object
- **Structure**:

```
{
  "functional_requirements": [...],
  "non_functional_requirements": [...],
  "business_rules": [...],
  "action_items": [...],
  "decisions": [...],
  "stakeholders": [...]
}
```

- **Validation**: JSON schema validation

Token Usage Estimates

Transcription

- **Local Whisper**: No tokens (local processing)
- **OpenAI Whisper API**: No tokens (separate pricing)

Requirements Extraction

- **Input**: ~500-2000 tokens per chunk
- **Output**: ~200-800 tokens per chunk
- **Total**: ~700-2800 tokens per chunk

Example: 100-message transcript (2 chunks)

- Input: ~2000 tokens
- Output: ~800 tokens
- Total: ~2800 tokens
- Cost (gpt-4o-mini): ~$0.42

Memory Requirements

Local Whisper

- **Base Model**: ~1 GB RAM
- **Small Model**: ~2 GB RAM
- **Medium Model**: ~5 GB RAM
- **Large Model**: ~10 GB RAM

Ollama

- **llama3.2 (3B)**: ~4 GB RAM/VRAM
- **mistral (7B)**: ~6 GB RAM/VRAM
- **llama3 (8B)**: ~8 GB RAM/VRAM

Processing Time Estimates

Small File (5 minutes audio, 50 messages)

- **Transcription (Local)**: ~5 minutes
- **Transcription (API)**: ~30 seconds
- **Extraction (GPT)**: ~5 seconds

- **Extraction (Ollama)**: ~10-20 seconds
- **Total (Local)**: ~5-6 minutes
- **Total (API)**: ~35 seconds

Large File (60 minutes audio, 500 messages)

- **Transcription (Local)**: ~60 minutes
- **Transcription (API)**: ~6 minutes
- **Extraction (GPT)**: ~50 seconds (10 chunks)
- **Extraction (Ollama)**: ~2-5 minutes (10 chunks)
- **Total (Local)**: ~62-65 minutes
- **Total (API)**: ~7 minutes

Appendix

A. Model Version History

Whisper

- **v1**: Initial release (2022)
- **v2**: Improved accuracy (2023)
- **v3**: Latest version (2024)

GPT Models

- **gpt-3.5-turbo**: Legacy model
- **gpt-4**: Previous generation
- **gpt-4o**: Current generation (optimized)
- **gpt-4o-mini**: Smaller, faster variant

Ollama Models

- **llama3.2**: Latest (2024)
- **llama3**: Previous generation
- **mistral**: Alternative option

B. API Rate Limits

OpenAI Whisper API

- **Free Tier**: Limited requests
- **Paid Tier**: Based on usage
- **Rate Limits**: Vary by account tier

OpenAI GPT API

- **Rate Limits**: Based on account tier
- **Free Tier**: 3 requests/minute
- **Paid Tier**: Higher limits

Ollama

- **No Rate Limits**: Local processing
- **Concurrent Requests**: Limited by hardware

C. Error Handling

The system includes comprehensive error handling for:

- Model loading failures
- API connection errors
- Invalid responses
- JSON parsing errors
- Timeout handling
- Rate limit handling

Document History

Version	Date	Changes
1.0	Nov 2024	Initial documentation

End of Models Documentation