Video Rating Service

A FastAPI-based service that rates candidate videos against a master video using various similarity metrics.

Features

- Convert video to audio using ffmpeg
- Transcribe audio using OpenAI's Whisper (tiny.en model)
- Calculate video similarity scores based on:
  - Keyword coverage: Measures important terms shared between videos
  - Content similarity: Uses TF-IDF and cosine similarity
  - Filler word penalty: Detects and penalizes use of filler words
  - Semantic similarity: Uses spaCy word embeddings for meaning comparison
- S3 integration for video file handling
- Error handling and validation
- Prometheus metrics for monitoring

Prerequisites

- Python 3.8+
- FFmpeg installed on your system
- AWS credentials configured for S3 access
- 2GB+ RAM recommended for model loading
- Disk space for temporary video processing

Installation

1. Create a virtual environment:

```
python -m venv venv
source venv/bin/activate
```

2. Install dependencies:

```
pip install -r requirements.txt
python -m spacy download en_core_web_md
```

Configuration

1. Create .env file with AWS credentials:

```
AWS_ACCESS_KEY_ID=your_key_id
AWS_SECRET_ACCESS_KEY=your_secret_key
AWS_DEFAULT_REGION=your_region
AWS_BUCKET=your_bucket_name
```

```
AWS_ENDPOINT=your_endpoint_url
AWS_USE_PATH_STYLE_ENDPOINT=true
```

2. Configure API settings in app/config.py:

- API_RATE_LIMIT: Requests per minute (default: 100)
- MAX_UPLOAD_SIZE: Maximum video file size (default: 500MB)
- ALLOWED_VIDEO_TYPES: Supported video formats
- REQUEST_TIMEOUT: API timeout in seconds

API Endpoints

1. Health Check

```
GET /health
```

Response:

```json
{
  "status": "healthy",
  "timestamp": "2023-11-09T12:00:00Z"
}
```

2. Upload Video

Upload a video file to S3.

```
POST /upload_video/
Content-Type: multipart/form-data
```

Request: - file: Video file (supported formats: mp4, mpeg, mov, avi, wmv)

Response:

```json
{
  "url": "s3://bucket/uuid-filename.mp4"
}
```

Errors: - 400: Invalid file extension/type, file size too large - 500: Upload failed

3. Rate Videos

Compare candidate videos against a master video.

```
POST /rate_videos/
Content-Type: application/json
```

Request:

```json
{
  "master_url": "s3://bucket/master.mp4",
  "candidate_urls": [
    "s3://bucket/candidate1.mp4",
    "s3://bucket/candidate2.mp4"
  ]
}
```

Response:

```
{
  "results": [
    {
      "url": "s3://bucket/candidate1.mp4",
      "keyword_coverage": 0.85,
      "content_similarity": 0.78,
      "filler_word_penalty": 0.95,
      "semantic_similarity": 0.82,
      "aggregate_score": 0.84,
      "error": null
    },
    {
      "url": "s3://bucket/candidate2.mp4",
      "keyword_coverage": 0.0,
      "content_similarity": 0.0,
      "filler_word_penalty": 0.0,
      "semantic_similarity": 0.0,
      "aggregate_score": 0.0,
      "error": "Error message if processing failed",
      "error_code": "PROCESSING_ERROR"
    }
  ]
}
```

Score meanings: - keyword_coverage (0-1): Higher means more key terms matched - content_similarity (0-1): Higher means more similar content - filler_word_penalty (0-1): Lower means more filler words present (penalized score) - semantic_similarity (0-1): Higher means more similar meaning - aggregate_score (0-1): Weighted average of all metrics

Notes: - All requested candidates will appear in results, even if processing fails. - Check `error` and `error_code` in response for specific failure reasons. If `error` is null, the processing was successful.

4. Readiness Check

`GET /readiness`

Response:

```
{
  "status": "ready",
  "checks": {
    "s3": "ok",
    "ffmpeg": "ok",
    "whisper": "ok"
  }
}
```

Example Usage

1. Upload videos:

```
# Upload master video
curl -X POST "http://localhost:8000/upload_video/" \
     -H "Content-Type: multipart/form-data" \
     -F "file=@master.mp4"

# Upload candidate video
curl -X POST "http://localhost:8000/upload_video/" \
     -H "Content-Type: multipart/form-data" \
     -F "file=@candidate.mp4"
```

2. Rate videos:

```
curl -X POST "http://localhost:8000/rate_videos/" \
     -H "Content-Type: application/json" \
     -d '{
       "master_url": "s3://bucket/master.mp4",
       "candidate_urls": [
         "s3://bucket/candidate1.mp4",
         "s3://bucket/candidate2.mp4"
       ]
     }'
```

Error Handling

The service handles various error cases: - Invalid S3 URLs - Video download failures - Audio conversion issues - Transcription errors - File size/format validation - Resource availability - API rate limiting

Errors are returned with appropriate HTTP status codes and descriptive messages in JSON format:

```
{
  "detail": "Error description",
  "request_id": "uuid",
  "error_code": "ERROR_TYPE"
}
```

Development

Start the development server:

```
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

Access API documentation: - Swagger UI: http://localhost:8000/docs

Production Deployment

For production: 1. Use proper SSL/TLS termination 2. Configure appropriate timeouts 3. Set up monitoring using provided Prometheus metrics 4. Consider using a task queue for long-running operations 5. Implement appropriate authentication/authorization 6. Configure proper logging and error tracking