EasyRec Online - API Usage Examples

This document provides examples of how to use the **EasyRec Online** REST API, which extends Alibaba's EasyRec framework with real-time learning capabilities.

🏗️ Architecture Overview

EasyRec Online = **Alibaba EasyRec** (Core) + **Online Learning Extensions** (This Project)

- 🔵 **Alibaba EasyRec**: Model training, evaluation, configuration format
- 🟢 **EasyRec Online**: REST API, streaming support, incremental updates

Starting the API Server

Local Development

```
# Install dependencies and setup
bash setup.sh

# Start the server
python api/app.py
```

Using Docker

```
# Build and run with Docker Compose
docker-compose up --build
```

Using Gunicorn (Production)

```
# Start with Gunicorn
python scripts/serve.py
```

API Endpoints

Core Recommendation Endpoints (🔵 Based on Alibaba EasyRec)

1. Health Check

Endpoint: `GET /health`

Description: Check if the API is running and get model status.

Example:

```
curl -X GET http://localhost:5000/health
```

Response:

```
{
  "status": "healthy",
  "message": "EasyRec Online API is running",
  "model_status": {
    "model_dir": "models/checkpoints/deepfm_movies",
    "config_path": "config/deepfm_config.prototxt",
    "model_loaded": true,
    "model_type": "DeepFM",
    "embedding_dim": 32,
    "status": "ready"
  },
  "features": {
    "online_learning": true,
    "streaming_support": true,
    "incremental_updates": true
  }
}
```

2. Model Information

Endpoint: `GET /model/info`

Description: Get detailed information about the loaded model.

Example:

```
curl -X GET http://localhost:5000/model/info
```

3. Predict Scores

Endpoint: `POST /predict`

Description: Get prediction scores for user-item pairs.

Request Body:

```
{
  "user_ids": [123, 456, 789],
  "item_ids": [1, 2, 3]
}
```

Example:

```
curl -X POST http://localhost:5000/predict \
  -H "Content-Type: application/json" \
  -d '{
    "user_ids": [123, 456, 789],
    "item_ids": [1, 2, 3]
  }'
```

Response:

```
{
  "success": true,
  "data": {
    "predictions": [
      {"user_id": 123, "item_id": 1, "score": 0.75},
      {"user_id": 456, "item_id": 2, "score": 0.82},
      {"user_id": 789, "item_id": 3, "score": 0.64}
    ],
    "count": 3
  }
}
```

4. Get Recommendations

Endpoint: `POST /recommend`

Description: Get top-k item recommendations for a user.

Request Body:

```
{
  "user_id": 123,
  "candidate_items": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
  "top_k": 5
}
```

Example:

```
curl -X POST http://localhost:5000/recommend \
  -H "Content-Type: application/json" \
  -d '{
    "user_id": 123,
    "candidate_items": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
```

```
    "top_k": 5
  }'
```

Response:

```
{
  "success": true,
  "data": {
    "user_id": 123,
    "recommendations": [
      {"item_id": 7, "score": 0.89, "rank": 1},
      {"item_id": 3, "score": 0.84, "rank": 2},
      {"item_id": 10, "score": 0.81, "rank": 3},
      {"item_id": 1, "score": 0.78, "rank": 4},
      {"item_id": 5, "score": 0.72, "rank": 5}
    ],
    "count": 5
  }
}
```

5. Get User Embedding

Endpoint: `GET /embeddings/user/{user_id}`

Description: Get the embedding vector for a specific user.

Example:

```
curl -X GET http://localhost:5000/embeddings/user/123
```

Response:

```
{
  "success": true,
  "data": {
    "user_id": 123,
    "embedding": [0.1, -0.2, 0.3, ..., 0.05],
    "dimension": 32
  }
}
```

6. Get Item Embedding

Endpoint: `GET /embeddings/item/{item_id}`

Description: Get the embedding vector for a specific item.

Example:

```
curl -X GET http://localhost:5000/embeddings/item/456
```

Response:

```
{
  "success": true,
  "data": {
    "item_id": 456,
    "embedding": [-0.1, 0.4, -0.2, ..., 0.15],
    "dimension": 32
  }
}
```

Online Learning Endpoints (🟢 EasyRec Online Extensions)

7. Add Training Data

Endpoint: `POST /online/data/add`

Description: Add training samples for incremental learning.

Request Body:

```
{
  "samples": [
    {
      "user_id": 123,
      "item_id": 456,
      "label": 1,
      "rating": 4.5,
      "features": {
        "gender": 1,
        "age": 3,
        "genres": "Action|Drama"
      }
    }
  ]
}
```

Example:

```
curl -X POST http://localhost:5000/online/data/add \
  -H "Content-Type: application/json" \
  -d '{
    "samples": [
      {"user_id": 123, "item_id": 456, "label": 1, "rating": 4.5}
    ]
  }'
```

8. Start Incremental Training

Endpoint: `POST /online/training/start`

Description: Start real-time incremental training with streaming data.

Request Body:

```
{
  "kafka_config": {
    "servers": "localhost:9092",
    "topic": "easyrec_training",
    "group": "easyrec_online",
    "offset_time": "20240101 00:00:00"
  },
  "update_config": {
    "dense_save_steps": 100,
    "sparse_save_steps": 100
  }
}
```

Example:

```
curl -X POST http://localhost:5000/online/training/start \
  -H "Content-Type: application/json" \
  -d '{
    "kafka_config": {
      "servers": "localhost:9092",
      "topic": "easyrec_training",
      "group": "easyrec_online"
    }
  }'
```

9. Get Training Status

Endpoint: `GET /online/training/status`

Description: Get current training status and progress.

Example:

```
curl -X GET http://localhost:5000/online/training/status
```

Response:

```
{
  "success": true,
  "data": {
    "is_training": true,
    "model_dir": "models/online/deepfm_movies",
    "latest_checkpoint": "models/online/deepfm_movies/model.ckpt-1500",
    "num_checkpoints": 15,
    "process_id": 12345
  }
}
```

10. Trigger Model Retraining

Endpoint: `POST /online/model/retrain`

Description: Trigger full or incremental model retraining.

Request Body:

```
{
  "retrain_type": "incremental",
  "export_after": true
}
```

Example:

```
curl -X POST http://localhost:5000/online/model/retrain \
  -H "Content-Type: application/json" \
  -d '{
    "retrain_type": "full",
    "export_after": true
  }'
```

11. Get Streaming Status

Endpoint: `GET /online/streaming/status`

Description: Get status of streaming data consumption.

Example:

```
curl -X GET http://localhost:5000/online/streaming/status
```

Response:

```
{
  "success": true,
  "data": {
    "connected": true,
    "running": false,
    "topic": "easyrec_training",
    "group": "easyrec_online",
    "current_offsets": {"0": 12345, "1": 12350}
  }
}
```

Python Client Example

```
import requests
import json

# API base URL
BASE_URL = "http://localhost:5000"

class EasyRecClient:
    def __init__(self, base_url=BASE_URL):
        self.base_url = base_url

    def health_check(self):
        """Check API health"""
        response = requests.get(f"{self.base_url}/health")
        return response.json()

    def predict_scores(self, user_ids, item_ids):
        """Predict scores for user-item pairs"""
        payload = {
            "user_ids": user_ids,
            "item_ids": item_ids
        }
        response = requests.post(
            f"{self.base_url}/predict",
            json=payload
        )
        return response.json()

    def get_recommendations(self, user_id, candidate_items, top_k=10):
```

```
        """Get recommendations for a user"""
        payload = {
            "user_id": user_id,
            "candidate_items": candidate_items,
            "top_k": top_k
        }
        response = requests.post(
            f"{self.base_url}/recommend",
            json=payload
        )
        return response.json()

    def get_user_embedding(self, user_id):
        """Get user embedding"""
        response = requests.get(f"
{self.base_url}/embeddings/user/{user_id}")
        return response.json()

    # Online Learning Methods (EasyRec Online Extensions)

    def add_training_data(self, samples):
        """Add training samples for incremental learning"""
        payload = {"samples": samples}
        response = requests.post(
            f"{self.base_url}/online/data/add",
            json=payload
        )
        return response.json()

    def start_incremental_training(self, kafka_config=None,
update_config=None):
        """Start incremental training"""
        payload = {}
        if kafka_config:
            payload["kafka_config"] = kafka_config
        if update_config:
            payload["update_config"] = update_config

        response = requests.post(
            f"{self.base_url}/online/training/start",
            json=payload
        )
        return response.json()

    def get_training_status(self):
        """Get training status"""
        response = requests.get(f"
{self.base_url}/online/training/status")
        return response.json()

    def trigger_retraining(self, retrain_type="incremental",
export_after=True):
        """Trigger model retraining"""
```

```
        payload = {
            "retrain_type": retrain_type,
            "export_after": export_after
        }
        response = requests.post(
            f"{self.base_url}/online/model/retrain",
            json=payload
        )
        return response.json()

    def get_streaming_status(self):
        """Get streaming status"""
        response = requests.get(f"
{self.base_url}/online/streaming/status")
        return response.json()

# Usage example
if __name__ == "__main__":
    client = EasyRecClient()

    # Check health
    health = client.health_check()
    print("Health:", health)

    # Get recommendations
    recommendations = client.get_recommendations(
        user_id=123,
        candidate_items=[1, 2, 3, 4, 5],
        top_k=3
    )
    print("Recommendations:", recommendations)

    # Online Learning Examples

    # Add new training data
    new_samples = [
        {"user_id": 123, "item_id": 6, "label": 1, "rating": 4.8},
        {"user_id": 124, "item_id": 7, "label": 0, "rating": 2.1}
    ]
    result = client.add_training_data(new_samples)
    print("Added training data:", result)

    # Check training status
    status = client.get_training_status()
    print("Training status:", status)

    # Start incremental training (if not already running)
    if not status.get('data', {}).get('is_training', False):
        training_result = client.start_incremental_training()
        print("Started training:", training_result)

    # Check streaming status
```

```
    streaming_status = client.get_streaming_status()
    print("Streaming status:", streaming_status)
```

JavaScript Client Example

```
class EasyRecClient {
    constructor(baseUrl = 'http://localhost:5000') {
        this.baseUrl = baseUrl;
    }

    async healthCheck() {
        const response = await fetch(`${this.baseUrl}/health`);
        return await response.json();
    }

    async predictScores(userIds, itemIds) {
        const response = await fetch(`${this.baseUrl}/predict`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                user_ids: userIds,
                item_ids: itemIds
            })
        });
        return await response.json();
    }

    async getRecommendations(userId, candidateItems, topK = 10) {
        const response = await fetch(`${this.baseUrl}/recommend`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                user_id: userId,
                candidate_items: candidateItems,
                top_k: topK
            })
        });
        return await response.json();
    }
}

// Usage example
const client = new EasyRecClient();

client.getRecommendations(123, [1, 2, 3, 4, 5], 3)
```

```
    .then(result => console.log('Recommendations:', result))
    .catch(error => console.error('Error:', error));
```

Error Handling

All endpoints return a consistent error format:

```
{
  "success": false,
  "error": "Error message describing what went wrong"
}
```

Common HTTP status codes:

- `200`: Success
- `400`: Bad Request (invalid input)
- `404`: Not Found
- `500`: Internal Server Error