E-commerce Semantic Search API Documentation

Overview

The E-commerce Semantic Search API provides intelligent product search capabilities using advanced semantic search technology powered by AWS Bedrock Titan. This API allows you to search through product catalogs using natural language queries and supports various filtering options.

Key Features: - Natural language processing for intuitive search queries - Semantic understanding beyond keyword matching - Real-time vector embeddings using AWS Bedrock Titan - Advanced filtering and pagination support

Base URL

```
https://9ahb9n78o3.execute-api.us-east-1.amazonaws.com/prod/search
```

Authentication

Currently, this API does not require authentication. However, please ensure you follow the rate limiting guidelines provided below.

Endpoint Details

POST /search

Performs a semantic search across the product catalog.

Method: POST **Content-Type:** application/json

Request Format

Request Body Schema

```
{
  "query": "string (required)",
  "limit": "integer (optional, default: 10)",
  "offset": "integer (optional, default: 0)",
  "filters": {
    "category": "string (optional)",
    "brand": "string (optional)",
    "priceMin": "number (optional)",
    "priceMax": "number (optional)"
  }
}
```

Field Descriptions

Field	Type	Required	Description
query	string	Yes	**Natural language search query** - Use conversational phrases like “comfortable running shoes for marathon training” or “wireless headphones with noise cancellation”
limit	integer	No	Number of results to return (default: 10, max: 100)
offset	integer	No	Number of results to skip for pagination (default: 0)
filters.category	string	No	Filter by product category
filters.brand	string	No	Filter by brand name
filters.priceMin	number	No	Minimum price filter
filters.priceMax	number	No	Maximum price filter

Semantic Query Examples

The API understands natural language queries and provides contextually relevant results. Here are examples of effective semantic search queries:

1. Athletic & Sports Queries

```
{
  "query": "comfortable running shoes for marathon training",
  "limit": 5
}
```

```
{
  "query": "lightweight athletic footwear for jogging",
  "limit": 5
}
```

```
{
  "query": "waterproof hiking boots for outdoor adventures",
  "limit": 5
}
```

2. Electronics & Technology Queries

```
{
  "query": "wireless noise-canceling headphones for travel",
  "limit": 5
}
```

```
{
  "query": "gaming laptop with high performance graphics",
  "limit": 5
}
```

```
{
  "query": "smartphone with excellent camera quality",
  "limit": 5
}
```

3. Fashion & Lifestyle Queries

```
{
  "query": "elegant dress for formal occasions",
  "limit": 5
}
```

```
{
  "query": "casual wear for weekend relaxation",
  "limit": 5
}
```

```
{
  "query": "winter jacket for cold weather protection",
  "limit": 5
}
```

4. Home & Kitchen Queries

```
{
  "query": "kitchen appliances for meal preparation",
  "limit": 5
}
```

```
{
  "query": "comfortable furniture for living room",
  "limit": 5
}
```

5. Intent-Based Queries

```
{
  "query": "gift ideas for fitness enthusiasts",
  "limit": 10
}
```

```
{
  "query": "products for home office setup",
  "limit": 10
}
```

```
{
  "query": "items for outdoor camping trip",
  "limit": 10
}
```

Response Format

Success Response Schema

```
{
  "products": [
    {
      "id": "string",
      "name": "string",
      "description": "string",
      "price": "number",
      "category": "string",
      "brand": "string",
      "imageUrl": "string",
      "score": "number"
    }
  ],
  "total": "number",
  "limit": "number",
  "offset": "number",
  "processingTimeMs": "number"
}
```

Response Field Descriptions

Field	Type	Description
products	array	Array of product objects matching the search criteria
products[].id	string	Unique product identifier
products[].name	string	Product title/name
products[].description	string	Product description
products[].price	number	Product price
products[].category	string	Product category
products[].brand	string	Product brand
products[].imageUrl	string	URL to product image
products[].score	number	Relevance score (higher = more relevant)
total	number	Total number of matching products
limit	number	Number of results returned in this response
offset	number	Number of results skipped
processingTimeMs	number	Time taken to process the request (in milliseconds)

Complete Request Examples

1. Basic Semantic Search

Request:

```
curl -X POST https://9ahb9n78o3.execute-api.us-east-1.amazonaws.com/
        prod/search \
  -H "Content-Type: application/json" \
  -d '{
    "query": "comfortable running shoes for marathon training"
  }'
```

2. Semantic Search with Filters

Request:

```
curl -X POST https://9ahb9n78o3.execute-api.us-east-1.amazonaws.com/
        prod/search \
  -H "Content-Type: application/json" \
  -d '{
    "query": "wireless headphones with noise cancellation",
    "limit": 10,
    "filters": {
      "brand": "Sony",
      "priceMax": 300
    }
  }'
```

3. Intent-Based Search with Pagination

Request:

```
curl -X POST https://9ahb9n78o3.execute-api.us-east-1.amazonaws.com/
        prod/search \
  -H "Content-Type: application/json" \
  -d '{
    "query": "gift ideas for fitness enthusiasts under budget",
    "limit": 20,
    "offset": 20,
    "filters": {
      "priceMax": 100
    }
  }'
```

4. Lifestyle-Oriented Search

Request:

```
curl -X POST https://9ahb9n78o3.execute-api.us-east-1.amazonaws.com/
        prod/search \
  -H "Content-Type: application/json" \
  -d '{
    "query": "products for creating a productive home office
        environment",
    "limit": 15
  }'
```

Response Examples

Successful Response


```
{
  "products": [
    {
      "id": "B08VRJ6F2Q",
      "name": "ASICS Men's GT-1000 10 Running Shoes",
      "description": "ENGINEERED MESH A stretch mesh upper adjusts to the shape
        of the foot and provides an excellent fit. LIGHTWEIGHT CUSHIONING Our
        FLYTEFOAM midsole technology is soft and lightweight...",
      "price": 89.99,
      "category": "[\"Clothing, Shoes & Jewelry\",\"Men\",\"Shoes\",\"Athletic\",
        \"Running\",\"Road Running\"]",
      "brand": "Asics",
      "imageUrl": "https://m.media-amazon.com/images/I/
        61wReiFg2mL.__AC_SX395_SY395_QL70_FMwebp_.jpg",
      "score": 23.85
    },
    {
      "id": "B01HD620I8",
      "name": "Salomon Men's Speedcross 4 Trail Running Shoes",
      "description":
        "The 4th edition of our iconic and aggressively lugged trail runner for
        tearing through technical, soft ground with speed...",
      "price": 102.46,
      "category": "[\"Clothing, Shoes & Jewelry\",\"Men\",\"Shoes\",\"Athletic\",
        \"Running\",\"Trail Running\"]",
      "brand": "Salomon",
      "imageUrl": "https://m.media-amazon.com/images/I/71-
        h7s7wWlL.__AC_SY395_SX395_QL70_FMwebp_.jpg",
      "score": 21.24
    }
  ],
  "total": 354,
  "limit": 5,
  "offset": 0,
  "processingTimeMs": 52
}
```

Best Practices for Semantic Queries

1. Use Natural Language

✅ **Good:** “comfortable running shoes for marathon training”

❌ **Avoid:** “shoes running marathon comfortable”

2. Include Context and Intent

✅ **Good:** “wireless headphones for travel with noise cancellation”

❌ **Avoid:** “headphones wireless”

3. Specify Use Cases

✅ **Good:** “kitchen appliances for meal preparation”

✅ **Good:** “gift ideas for fitness enthusiasts”

✅ **Good:** “products for home office setup”

4. Combine Descriptive Terms

✅ **Good:** “waterproof hiking boots for outdoor adventures”

✅ **Good:** “lightweight athletic wear for summer workouts”

5. Express Specific Needs

✅ **Good:** “ergonomic office chair for long work sessions”

✅ **Good:** “budget-friendly smartphone with good camera”

Error Responses

400 Bad Request

Cause: Invalid request format or missing required fields

```
{
  "error": "Bad Request",
  "message": "Missing required field: query",
  "statusCode": 400
}
```

500 Internal Server Error

Cause: Server-side error during processing

```
{
  "error": "Internal Server Error",
  "message": "Failed to perform semantic search",
  "statusCode": 500
}
```

Rate Limiting

To ensure fair usage and optimal performance:

- **Rate Limit:** 100 requests per minute per IP address
- **Burst Limit:** 10 requests per second
- **Daily Limit:** 10,000 requests per day per IP address

When rate limits are exceeded, you'll receive a `429 Too Many Requests` response.

Integration Examples

JavaScript/Node.js

```
const searchProducts = async (query, filters = {}) => {
  const apiUrl = 'https://9ahb9n78o3.execute-api.us-east-1.amazonaws.com/prod/
        search';

  const requestBody = {
    query,
    limit: 10,
    offset: 0,
    filters
  };

  try {
    const response = await fetch(apiUrl, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(requestBody)
    });

    if (!response.ok) {
      throw new Error(`HTTP ${response.status}: ${response.statusText}`);
    }

    return await response.json();
  } catch (error) {
    console.error('Search API error:', error);
    throw error;
  }
};

// Usage examples
const results1 = await searchProducts('comfortable running shoes for marathon
        training');
const results2 = await searchProducts('wireless headphones for travel', {
        priceMax: 200 });
const results3 = await searchProducts('gift ideas for fitness enthusiasts', {
        priceMax: 100 });
```

Python

```python
import requests
import json

def search_products(query, filters=None, limit=10, offset=0):
    url = 'https://9ahb9n78o3.execute-api.us-east-1.amazonaws.com/prod/search'

    payload = {
        'query': query,
        'limit': limit,
        'offset': offset
    }

    if filters:
        payload['filters'] = filters

    headers = {
        'Content-Type': 'application/json'
    }

    try:
        response = requests.post(url, json=payload, headers=headers)
        response.raise_for_status()
        return response.json()
    except requests.exceptions.RequestException as e:
        print(f"Search API error: {e}")
        raise

# Usage examples
results1 = search_products('comfortable running shoes for marathon training')
results2 = search_products('wireless headphones for travel', {'priceMax': 200})
results3 = search_products('home office furniture for productivity',
        {'category': 'Furniture'})
```

Performance

- **Average Response Time:** 50-150ms (with semantic processing)
- **Peak Response Time:** < 2 seconds
- **Availability:** 99.9% uptime SLA
- **Geographic Distribution:** Optimized for US East region

Troubleshooting

Common Issues

1. **Empty Results**
   - Try broader, more natural language terms
   - Remove overly specific filters
   - Use intent-based queries (e.g., “gift ideas for…”)
2. **Slow Response Times**
   - Reduce result limit for faster responses
   - Implement client-side caching for repeated queries
   - Consider pagination for large result sets
3. **Rate Limiting**
   - Implement exponential backoff for retries
   - Cache results when possible
   - Batch multiple searches if needed

Changelog

Version 1.1.0 (Current)

- Enhanced semantic search with AWS Bedrock Titan embeddings

- Improved natural language understanding
- Better contextual relevance scoring
- Added intent-based search capabilities

Version 1.0.0

- Initial release of semantic search API
- Support for text-based and vector-based search
- Category, brand, and price filtering
- Pagination support
- Rate limiting implementation

Document Version: 1.1.0 **Last Updated:** October 1, 2025 **API Version:** v1