E-commerce Semantic Search API Documentation

Overview

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

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

```json
{
  "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	The search query in natural language (e.g., “wireless headphones”, “summer dresses”, “gaming laptop”)
`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

Response Format

Success Response Schema

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

Response Field Descriptions

Field	Type	Description
`products`	array	Array of product objects matching the search criteria
`products[].id`	string	Unique product identifier
`products[].title`	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
`processingTime`	number	Time taken to process the request (in milliseconds)

Request Examples

1. Basic Search

Request:

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

2. 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": "laptop",
    "limit": 20,
    "offset": 20
  }'
```

3. 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": "smartphone",
    "limit": 10,
    "offset": 0,
    "filters": {
      "brand": "Apple",
      "priceMin": 500,
      "priceMax": 1200,
      "category": "Electronics"
    }
  }'
```

4. Category-only Search

Request:

```
curl -X POST https://9ahb9n78o3.execute-api.us-east-1.amazonaws.com/prod/search \
  -H "Content-Type: application/json" \
  -d '{
    "query": "",
    "limit": 15,
    "filters": {
      "category": "Clothing"
    }
  }'
```

Response Examples

Successful Response

```json
{
  "products": [
    {
      "id": "prod_12345",
      "title": "Sony WH-1000XM4 Wireless Noise Canceling Headphones",
      "description": "Industry-leading noise canceling with Dual Noise Sensor
        technology",
      "price": 299.99,
      "category": "Electronics",
      "brand": "Sony",
      "imageUrl": "https://example.com/images/sony-wh1000xm4.jpg",
      "score": 0.95
    },
    {
      "id": "prod_67890",
      "title": "Apple AirPods Pro (2nd Generation)",
      "description": "Active Noise Cancellation, Transparency mode, Spatial audio",
      "price": 249.99,
      "category": "Electronics",
      "brand": "Apple",
      "imageUrl": "https://example.com/images/airpods-pro-2.jpg",
      "score": 0.87
    }
  ],
  "total": 45,
  "limit": 10,
  "offset": 0,
  "processingTime": 234
}
```

Error Responses

400 Bad Request

Cause: Invalid request format or missing required fields

Response:

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

500 Internal Server Error

Cause: Server-side error during processing

Response:

```json
{
  "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.

Best Practices

1. Query Optimization

- **Use natural language:** "comfortable running shoes for women" instead of "shoes women running"
- **Be specific:** Include relevant details like size, color, or features
- **Avoid excessive keywords:** Focus on meaningful search terms

2. Pagination

- Use `limit` and `offset` for efficient pagination
- Recommended page size: 10-20 items
- For large datasets, consider using cursor-based pagination

3. Filtering

- Combine text search with filters for precise results
- Use price ranges for better user experience

• Category filters help narrow down large result sets

4. Error Handling

Always implement proper error handling in your client code:

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

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

  const data = await response.json();
  // Process successful response
} catch (error) {
  console.error('Search failed:', error);
  // Handle error appropriately
}
```

Integration Examples

JavaScript/Node.js

```javascript
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
const results = await searchProducts('wireless earbuds', {
  brand: 'Sony',
  priceMax: 200
});
```

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
results = search_products('gaming laptop', {
    'category': 'Electronics',
    'priceMin': 800,
    'priceMax': 2000
})
```

PHP

```php
<?php
function searchProducts($query, $filters = [], $limit = 10, $offset = 0) {
    $url = 'https://9ahb9n78o3.execute-api.us-east-1.amazonaws.com/prod/search';

    $data = [
        'query' => $query,
        'limit' => $limit,
        'offset' => $offset
    ];

    if (!empty($filters)) {
        $data['filters'] = $filters;
    }

    $options = [
        'http' => [
            'header' => "Content-Type: application/json\r\n",
            'method' => 'POST',
            'content' => json_encode($data)
        ]
    ];

    $context = stream_context_create($options);
    $result = file_get_contents($url, false, $context);

    if ($result === FALSE) {
        throw new Exception('Search API request failed');
    }

    return json_decode($result, true);
}

// Usage
$results = searchProducts('bluetooth speaker', [
    'brand' => 'JBL',
    'priceMax' => 150
]);
?>
```

Response Time and Performance

- **Average Response Time:** 200-500ms
- **Peak Response Time:** < 2 seconds
- **Availability:** 99.9% uptime SLA
- **Geographic Distribution:** Optimized for US East region

Troubleshooting

Common Issues

1. **Empty Results**
   - Check query spelling and terminology
   - Try broader search terms
   - Remove or adjust filters

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 1.0.0 (Current)

- 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: 1.0.0 **Last Updated:** October 1, 2025 **API Version:** v1