搜索中台技术培训

深入理解搜索系统的技术架构与优化策略

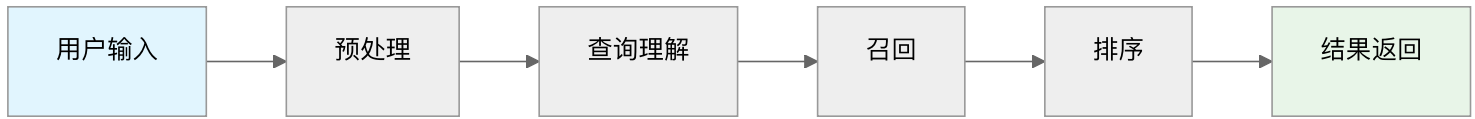
开始 →

培训目录

1. 搜索中台技术培训
2. 培训目录
3. 第一章：搜索过程概览
4. 第二章：搜索评分体系
5. 第三章：搜索粗排和精排
6. 第四章：搜索设置解释及调整方法
7. 培训总结
8. 谢谢观看！

第一章：搜索过程概览

搜索系统整体架构



核心流程

- 预处理：输入过滤、分词、规范化
- 查询理解：实体识别、同义词扩展
- 召回：多策略召回、粗排
- 排序：精排、个性化
- 返回：分页、结果优化

关键组件

- Elasticsearch：核心搜索引擎
- Prisma ORM：数据库操作
- 缓存系统：多层缓存优化
- 日志系统：性能监控

1.1 预处理阶段

主要功能

- **输入过滤**: 去除禁用词、特殊字符
- **分词处理**: 动态词典分词、新词识别
- **查询规范化**: 去除括号、标准化格式

处理示例

输入过滤

```
"apple苹果手机！！！" → "apple苹果手机"
```

分词处理

```
"苹果手机" → ["苹果", "手机"]
```

查询规范化

```
"iPhone 15（Pro Max版本）" → "iPhone 15 Pro Max版本"
```

1.2 查询理解阶段

命名实体识别

识别品牌、类目、属性、标签等实体

```
input: "苹果手机"
output: {
  brand: "苹果",
  category: "手机"
}
```

精准匹配

直接匹配系统中已有的品牌和类目

```
input: "华为"
output: { brandId: "huawei", confidence: 1.0 }
```

同义词扩展

使用同义词库扩展未直接匹配的类目

```
input: "笔记本"
expanded: ["笔记本电脑", "便携式电脑", "laptop"]
```

1.3 评分与召回阶段

多因子联合评分

- 实体关系评分: 品牌、类目匹配度
- 用户偏好评分: 个人历史行为
- 平台热度评分: 商品受欢迎程度

计算示例:

```
"苹果手机" → 品牌权重: 2.0 × 类目权重: 1.5 = 3.5
```

ES粗排召回

BM25算法 + 业务权重

```
final_score = BM25: 12.5 + 语义增强: 3.2
            + 质量评分: 1.8 = 17.5
```

得分归一化

```
销量: 10000件 → 归一化: 0.85分
Math.log1p(score/k) / Math.log1p(xmax/k)
```

1.4 精排阶段

业务模型精排

多级排序逻辑：按优先级顺序比较，只有相等时才比较下一个维度

```
排序优先级: first → OBB → SEM → PQ → VSL → PR
```

排序示例对比

商品A

```
ES评分: 17.5
品牌加权: 2.0
语义增强: 3.2
————————————
优先级排序
```

商品B

```
ES评分: 17.5
品牌加权: 1.5
语义增强: 3.8
————————————
次要排序
```

完整搜索示例流程

用户输入: `"华为笔记本电脑"`

1. **预处理:** `"华为笔记本电脑"` → `["华为", "笔记本电脑"]`
2. **实体识别:** 品牌= `"华为"` , 类目= `"笔记本电脑"`
3. **评分计算:** 用户输入与品牌匹配度：与品牌“华为”匹配度评分： `1.0` 、与品牌“huawei”匹配度评分： `0.9` ..... 相应品牌的商品按匹配度加分 用户输入与类目匹配度：与类目“笔记本电脑”匹配度评分： `1.0` 、与类目“游戏本”匹配度评分： `0.9` 、与类目“轻薄本”匹配度评分： `0.9` 与类目“电脑”匹配度评分： `0.8` ..... 相应类目的商品按匹配度加分 BM25评分 `8.3` 计算商品标题得分
4. **召回结果:** 300件相关商品（动态缓存扩展）
5. **精排输出:** 按综合评分排序，返回前20件商品

第二章：搜索评分体系

评分体系架构

```
最终评分 = BM25评分 × 权重配置 + 语义增强评分 + 商品质量评分 + 品牌提升评分
```

核心算法

- **BM25基础评分**: TF-IDF改进版
- **语义增强评分**: 实体匹配加权
- **商品质量评分**: 销售数据评估
- **品牌提升评分**: 店铺信誉加权
- **归一化处理**: 使用对数缩放归一化到0,1区间，以便添加权重设置。
- **权重配置**: 动态调整评分比例

性能优化

- **缓存机制**: 减少重复计算
- **分层评分**: 粗排+精排结合

2.1 BM25基础评分

算法原理

BM25: TF-IDF的改进版本，考虑词频饱和度

归一化处理: 使用对数缩放归一化到0,1区间，以便添加权重设置。

```
// TitleBM25归一化脚本
double total = Math.log1p(_score/k)
               / Math.log1p(xmax/k);
return total * weight;
```

评分特点

- **词频饱和**: 避免单一词汇过度影响
- **文档长度归一化**: 公平对比不同长度文本
- **逆文档频率**: 稀有词汇获得更高权重
- **参数可调**: k1、b参数控制算法行为

```
BM25(q,d) = Σ IDF(qi) ×
            f(qi,d) × (k1 + 1) /
            (f(qi,d) + k1 × (1-b + b×|d|/avgdl))
```

参数定义:

- **qi**: 查询中的第i个词项
- **f(qi,d)**: 词项qi在文档d中的词频
- **|d|**: 文档d的长度
- **avgdl**: 文档集合的平均长度
- **k1**: 词频饱和参数 (通常1.2-2.0)
  - *控制词频的影响程度，值越大词频的影响越大*
- **b**: 长度归一化参数 (通常0.75)
  - *控制文档长度归一化的程度*
  - *b=0 完全忽略文档长度，b=1 完全归一化*

2.2 语义增强评分 (SemanticEnhancement)

权重分布

```
brand_weight: 2.0      // 品牌匹配权重
category_weight: 1.5   // 类目匹配权重
spec_weight: 1.0       // 规格匹配权重
```

计算公式

```
total = brand_score * brand_weight +
        catF_score * category_weight +
        catR_score * category_weight
```

核心思想

通过实体匹配为搜索结果提供语义相关性评分

不同类型的匹配具有不同的权重，一般类目匹配获得最高权重，可自由调整

2.2 语义增强评分 - 实际应用示例

计算示例：查询"苹果手机"

匹配结果

```
查询分析结果:
{
  brand: "苹果",      // 品牌识别成功
  category: "手机"    // 类目识别成功
}
```

评分计算

```
brand_score = 1.0     // 完全匹配
catF_score = 1.0      // 前置类目匹配
catR_score = 0.0      // 无后置类目

total = 1.0×2.0 + 1.0×1.5 + 0.0×1.5 = 3.5
```

应用场景对比

高匹配商品 (加分3.5)

- iPhone 15 Pro Max
- 苹果iPhone 14系列
- Apple手机配件

低匹配商品 (加分1.5)

- 华为手机 (仅类目匹配)
- 小米手机 (仅类目匹配)

无匹配商品 (加分0)

- 苹果电脑 (品牌匹配但类目不符)
- 三星平板 (品牌类目均不匹配)

语义增强效果: 相关商品获得更高评分，提升搜索精准度

2.3 商品质量评分 (ProductQuality)

销售数据权重分布

```
近30天销量:  60%权重  // 反映当前热度
近365天销量: 30%权重  // 反映稳定性
历史总销量:  10%权重  // 反映经典程度
```

计算公式

```
skuScore = (sku30 * 0.6) +
           (sku365 * 0.3) +
           (skuTotal * 0.1)
```

归一化处理

```
normalizedScore =
  Math.log1p(skuScore/k) /
  Math.log1p(xmax/k)
```

参数说明:

- `k`: 缩放因子 (通常为100)
- `xmax`: 最大期望值 (通常为1000)

实际示例

```
原始销量: 10000件
归一化后: 0.85分 (0-1区间)
```

2.4 其他评分组件

品牌提升评分 (OriginalBrandBoost)

店铺信誉评分:

- 官方店铺: +0.5分
- 旗舰店铺: +0.2分
- 品牌匹配: +1.0分

供应商星级评分 (VenderStarLevel)

- 1-5星评级: 直接使用星级作为权重
- 优质标签: "premium"标签额外加权

业务权重评分 (BusinessWeight)

测试模式权重:

```
testWeight.brand +
testWeight.product +
testWeight.shop +
testWeight.vender +
testWeight.categoryFront +
testWeight.categoryRaw
```

正式模式权重:

```
formalWeight.brand +
formalWeight.product +
formalWeight.shop +
formalWeight.vender +
formalWeight.categoryFront +
formalWeight.categoryRaw
```

第三章：搜索粗排和精排

两阶段排序架构

粗排阶段 (Elasticsearch层面)

- **Function Score查询**: 多维度评分合并
- **动态缓存扩展**: 基于缓存状态动态召回
- **SPU/SKU处理**: 去重聚合与属性统计
- **性能优化**: 分页优化与过滤条件

精排阶段 (应用层面)

- **排序器架构**: 多级排序组合
- **排序顺序配置**: 可配置优先级
- **分组排序策略**: 货架式排列
- **业务逻辑**: 个性化与推荐

3.1 粗排阶段 - Function Score查询

Function Score查询结构

```
{
  function_score: {
    query: { /* BM25基础查询 */ },
    functions: [
      { script_score: { script: semanticEnhancementScript } },
      { script_score: { script: originalBrandBoostScript } },
      { script_score: { script: productQualityScoreScript } },
      { script_score: { script: titleBM25NormalizerScript } }
    ],
    score_mode: "sum",      // 评分合并：相加
    boost_mode: "replace"  // 替换原始BM25评分
  }
}
```

动态缓存扩展策略

```
{
  expandSize: "max(需求数量 - 当前缓存, 0)",
  maxLimit: "单次查询不超过10000条",
  paginationEnd: "max(from + size, esRecallSize)"
}
```

3.2 SPU/SKU处理策略

SPU模式

去重聚合: 使用 `collapse` 功能按 `spuId` 去重

```
collapse: {
  field: "spuId",
  inner_hits: { name: "variants", size: 3 }
}
```

统计聚合: 获取真实SPU数量

```
aggs: {
  distinct_spu_count: {
    cardinality: { field: "spuId" }
  }
}
```

SKU模式

直接返回: 返回所有SKU结果，不进行去重

多SKU展示: 支持同一SPU下多个SKU的展示

```
// 同一商品的不同规格
{
  spuId: "iphone15",
  variants: [
    { skuId: "iphone15-128g-blue" },
    { skuId: "iphone15-256g-red" }
  ]
}
```

属性聚合: 收集可筛选的商品属性

```
aggs: {
  brands: { terms: { field: "brandName" } },
  categories: { terms: { field: "categoryName" } }
}
```

3.3 精排阶段 - 排序器架构

排序器映射表

```
const sorterMap = {
  first: byScore(),               // ES评分排序 - 主评分
  OBB: byOriginalBrandBoost(),    // 品牌提升排序 - 店铺权重
  SEM: bySemanticEnhancement(),   // 语义增强排序 - 实体匹配
  PQ: byProductQuality(),         // 商品质量排序 - 销售数据
  VSL: byVenderStarLevel(),       // 供应商星级 - 信誉评分
  PR: byPremium()                // 优质标签排序 - 高端商品
}
```

📋 默认排序配置

```
sorterOrder: "first,OBB,SEM,PQ,VSL,PR"
```

优先级解释:

1. **first**: ES综合评分（主排序依据）
2. **OBB**: 品牌店铺权重（提升官方店铺）
3. **SEM**: 语义匹配度（相关性优化）
4. **PQ**: 商品销售质量（市场表现）
5. **VSL**: 供应商信誉（服务质量）
6. **PR**: 优质商品标识（高端定位）

3.4 多级排序与分组策略

多级排序逻辑

```
function sorter(sortType, products, sorterOrder) {
  const order = sorterOrder.split(',');
  const comparators = order.map(key => sorterMap[key]);

  return products.sort((a, b) => {
    for (const comparator of comparators) {
      const result = comparator(a, b);
      if (result !== 0) return result;
    }
    return 0;  // 相等时保持原顺序
  });
}
```

分组排序策略

货架式排序: 先分组，组内按排序器配置排序

```
function divideGroup(list, compareFn, groupSize) {
  const groups = [];
  for (let i = 0; i < list.length; i += groupSize) {
    groups.push(list.slice(i, i + groupSize));
  }
  return groups.map(group =>
    group.sort(compareFn)
  );
}
```

优势:

- 提升品牌展示均衡性
- 避免头部商品垄断
- 增加用户选择多样性

第四章：搜索设置解释及调整方法

⚙️参数调优体系

权重配置

- 语义增强权重
- 脚本评分权重
- 业务权重设置
- 实体权重管理

归一化参数

- 对数缩放配置
- xmax/k参数调整
- 分数分布优化
- 曲线形状控制

性能优化

- 召回窗口配置
- 缓存策略设置
- 分页优化参数

调优原则: 先理解业务目标 → 分析数据分布 → 小步调整验证 → 持续监控反馈优化效果

4.1 权重配置调整

语义增强权重

```
brandWeight: 2.0        // 品牌匹配权重
categoryWeight: 1.5     // 类目匹配权重
resultWeight: 5.0       // 语义增强总权重
```

调整建议：

- 品牌导向：提高 `brandWeight` 到3.0
- 类目丰富：提高 `categoryWeight` 到2.0
- 降低干扰：减少 `resultWeight` 到3.0

其他脚本权重

```
titleBM25Weight: 5.0              // 标题BM25权重
originalBrandBoostWeight: 1.0     // 品牌店铺权重
productQualityWeight: 1.0         // 商品质量权重
```

权重平衡策略：

- 新平台：降低质量权重，突出相关性
- 成熟平台：提高质量权重，优化转化率

4.2 归一化参数配置

对数缩放配置

```
// 各评分组件的归一化参数
titleBM25Normalizer:    { xmax: 100, k: 10 }   // 标题BM25归一化
semanticEnhancement:    { xmax: 50,  k: 5 }    // 语义增强归一化
productQualityScore:    { xmax: 1000, k: 100 } // 商品质量归一化
```

参数说明与公式

参数定义:

- **xmax:** 最大期望值，用于归一化上限
- **k:** 缩放因子，控制曲线陡峭程度

归一化公式:

```
normalizedScore = Math.log1p(score/k) / Math.log1p(xmax/k)
```

调整策略

降低k值

效果： 增加低分商品权重
适用： 长尾商品曝光

提高xmax

4.3 召回配置与业务权重

召回配置

```
esRecallSize: 300                          // ES召回窗口大小
cardinalityThreshold: 40000                // 基数统计精度阈值
sorterOrder: "first,OBB,SEM,PQ,VSL,PR"   // 排序优先级
```

调整指南:

- 小流量: `esRecallSize: 150` 减少开销
- 大流量: `esRecallSize: 500` 提高精度
- 品牌优先: `"first,OBB,PQ,SEM,VSL,PR"`
- 销量优先: `"first,PQ,OBB,SEM,VSL,PR"`

业务权重调整

数据库实体权重:

```
-- 品牌权重调整
UPDATE Brand SET testValue = 2.0
WHERE brandId = 'target_brand';

-- 类目权重调整
UPDATE CategoryFront SET testValue = 1.5
WHERE categoryId = 'target_category';
```

权重调整工具:

- 搜索设置页面: `/project/[code]/search/setting`
- 实体库页面: `/project/[code]/entityLibrary`
- 权重范围: 0.1 - 5.0

4.4 性能优化与效果调整策略

性能优化配置

缓存策略

```
searchCache: TTL 5分钟     // 搜索结果缓存
sortCache: TTL 10分钟      // 排序结果缓存
esCache: TTL 1分钟         // ES查询缓存
```

缓存作用:

- 减少ES查询压力，提升响应速度
- 降低系统负载，提高并发处理能力
- 缓解热点查询对系统的冲击

A/B测试策略

权重对比测试

```
// 测试组配置
const testConfig = {
  semanticWeight: 3.0,     // vs 基线5.0
  brandBoostWeight: 2.0,   // vs 基线1.0
  qualityWeight: 0.5       // vs 基线1.0
};
```

排序策略对比

- 策略**A**: `"first,SEM,OBB,PQ"` (语义优先)
- 策略**B**: `"first,PQ,OBB,SEM"` (销量优先)
- 策略**C**: `"first,OBB,SEM,PQ"` (品牌优先)

测试指标: 点击率、转化率、用户满意度

培训总结

搜索技术核心要点回顾

系统架构

- 预处理 → 查询理解 → 召回 → 排序 → 返回
- Elasticsearch + 应用层精排的两阶段架构
- 多层缓存优化

评分体系

- BM25基础评分 + 多维度增强评分
- 语义增强、商品质量、品牌提升等组件
- 对数归一化处理与权重平衡

召回排序

- Function Score查询整合多脚本评分
- SPU/SKU处理策略与属性聚合
- 多级排序器与分组展示策略

参数调优

- 权重配置、归一化参数、召回窗口
- A/B测试验证与数据驱动优化
- 缓存策略优化

持续优化: 理解业务 → 数据分析 → 假设验证 → 监控反馈

谢谢观看！

搜索中台技术培训

为不断提高企业运营效率而奋战 · 三横科技