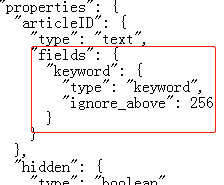
1. type=”text”

默认会设置两个field，一个是field本身，比如articleID，就是分词的；还有一个的话，就是field.keyword，articleID.keyword，默认不分词，会最多保留256个字符.

所以term filter，对text过滤，可以考虑使用内置的field.keyword来进行匹配。但是有个问题，默认就保留256个字符。所以尽可能还是自己去手动建立索引，指定not\_analyzed吧。在最新版本的es中，不需要指定not\_analyzed也可以，将type=keyword即可。

结论：type为text的字段需要建索引时指定为not\_analyzed（或者不用text用keyword类型），才能用term query

1. term和terms

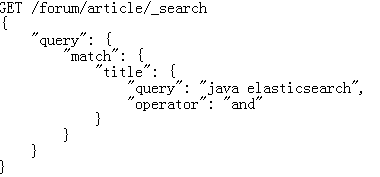
term对搜索文本不分词，直接拿去倒排索引中匹配，根据exact value进行搜索，数字、boolean、date天然支持

terms相当于sql中的in

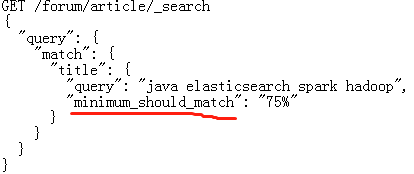
1. filter执行原理深度剖析（bitset机制与caching机制）
2. bool

可以包含must，must\_not，should，组合多个过滤条件。并且，bool可以嵌套。

1. match
   1. 进行full text全文检索.多个条件用空格分隔。如果要检索的field，是not\_analyzed类型的，那么match query也相当于term query.
   2. 如果你是希望所有的搜索关键字都要匹配的，那么就用and，可以实现单纯match query无法实现的效果

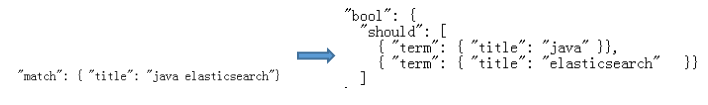
表示搜索的是要这两个字都有的

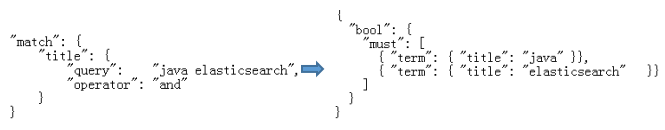
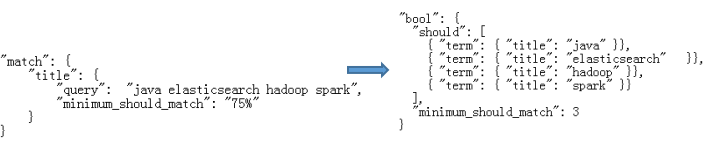
* 1. 指定一些关键字中，必须至少匹配其中的多少个关键字，才能作为结果返回



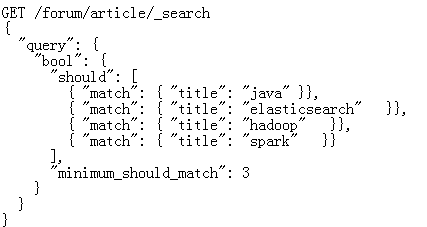
* 1. 底层实现

诸如上面的match query进行多值搜索的时候，es会在底层自动将这个match query转换为bool的语法，bool should，指定多个搜索词，同时使用term query



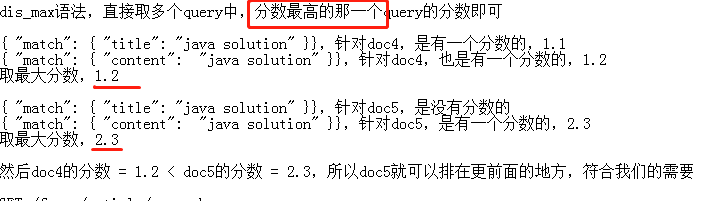
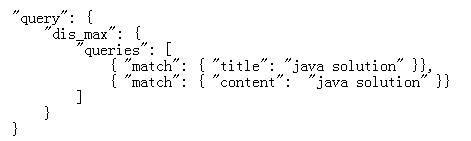
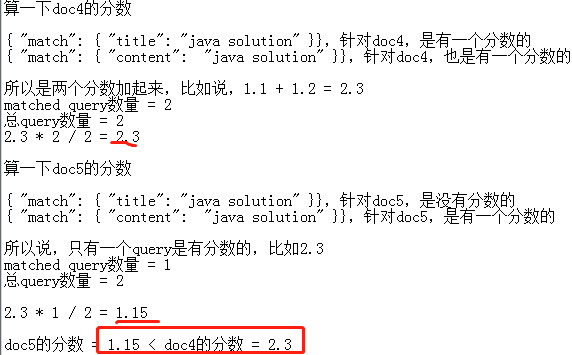
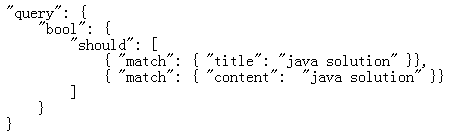
1. should
   1. 默认情况下，should是可以不匹配任何一个的。但是有个例外的情况，如果没有must的话，那么should中必须至少匹配一个才可以（第5.3条的另一种做法）



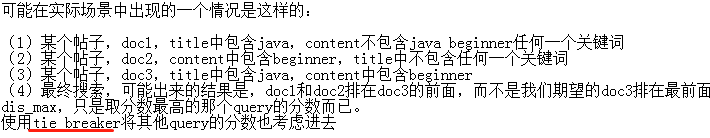
1. best fields策略
   1. best fields策略，就是说，将某一个field匹配尽可能多的关键词的doc优先返回回来.
   2. 实现方式：dis\_max语法，直接取多个query中，分数最高的那一个query的分数即可
   3. 举例说明：基础数据（两个对比）

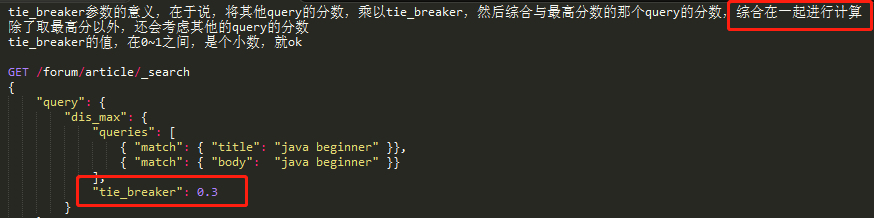
1547521024(1)1547520991(1)

默认情况下计算每个document的relevance score：每个query的分数，乘以matched query数量，除以总query数量。如果用dis\_max,那直接取分数最高的



* 1. tie\_breaker





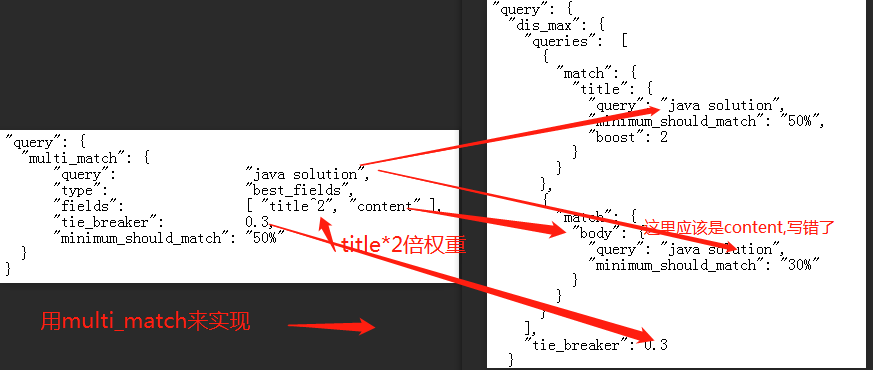
* 1. minimum\_should\_match

去长尾，long tail

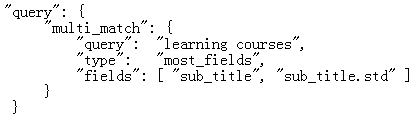
长尾，比如你搜索5个关键词，但是很多结果是只匹配1个关键词的，其实跟你想要的结果相差甚远，这些结果就是长尾

minimum\_should\_match，控制搜索结果的精准度，只有匹配一定数量的关键词的数据，才能返回

* 1. multi\_match语法实现dis\_max+tie\_breaker



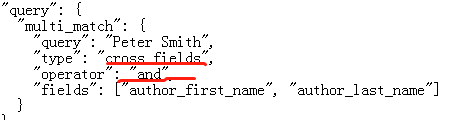
1. most fields策略
   1. 概念：尽可能返回更多field匹配到某个关键词的doc，优先返回回来
   2. 用法



1. cross fields策略
   1. 概念

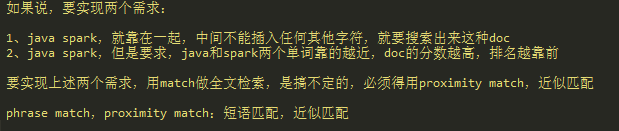
cross-fields搜索，一个唯一标识，跨了多个field。比如一个人，标识，是姓名；一个建筑，它的标识是地址。姓名可以散落在多个field中，比如first\_name和last\_name中，地址可以散落在country，province，city中。

* 1. 方法

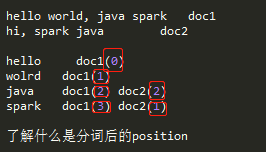


* 1. 优点
     1. 要求Peter必须在author\_first\_name或author\_last\_name中出现并且要求Smith必须在author\_first\_name或author\_last\_name中出现
     2. most\_fields，没办法用minimum\_should\_match去掉长尾数据，就是匹配的特别少的结果 --> 解决，既然每个term都要求出现，长尾肯定被去除掉了

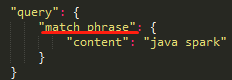
1. 近似匹配
   1. 概念



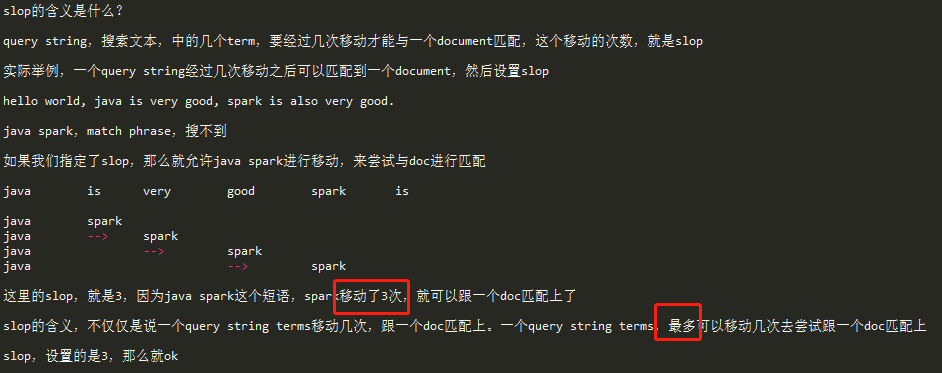
* 1. term position



* 1. match\_phrase
     1. 用法



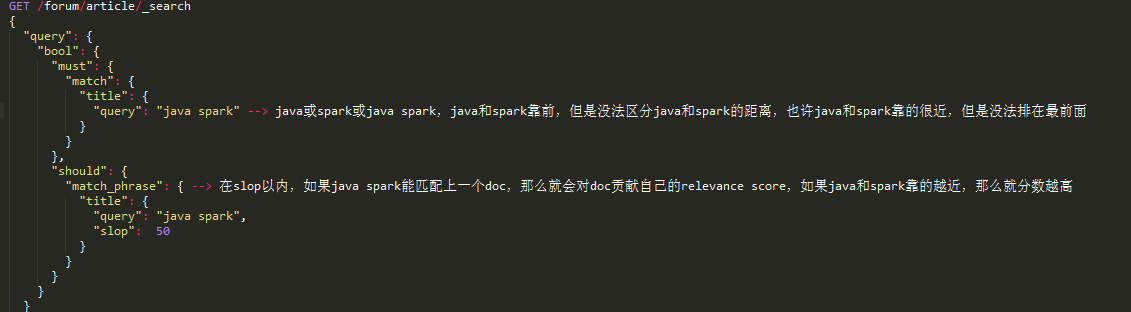
* + 1. 基本原理
       1. 要找到每个term都在的一个共有的那些doc，就是要求一个doc，必须包含每个term，才能拿出来继续计算
       2. 如果有两个单词，那后一个单词的position必须比前一个单词的position大1
  1. slop



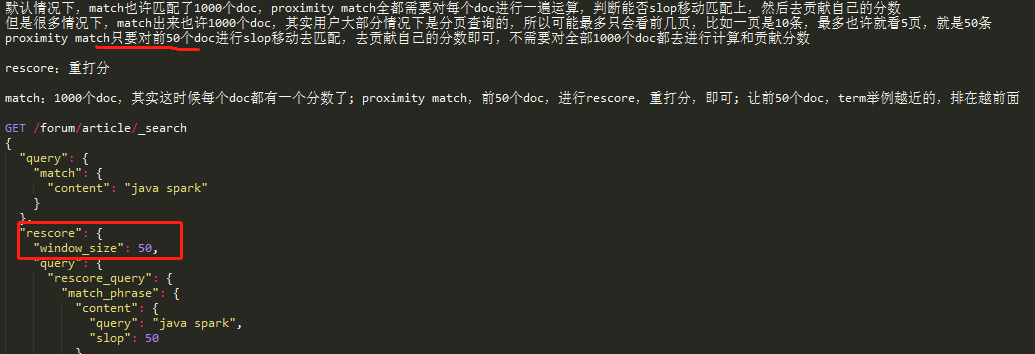


加了slop的phrase match，就是proximity match，近似匹配

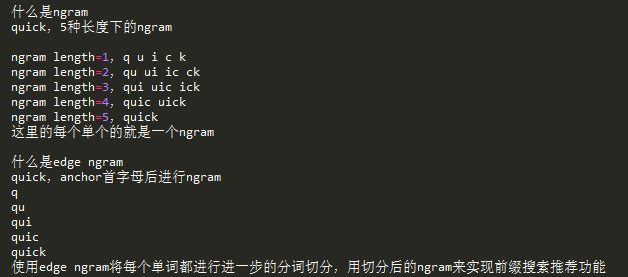
* 1. 使用match和近似匹配实现召回率与精准度的平衡
     1. 召回率:比如你搜索一个java spark，总共有100个doc，能返回多少个doc作为结果，就是召回率，recall
     2. 精准度:比如你搜索一个java spark，能不能尽可能让包含java spark，或者是java和spark离的很近的doc，排在最前面，precision
     3. 近似匹配的时候，召回率比较低，精准度太高了.但是有时可能我们希望的是匹配到几个term中的部分，就可以作为结果出来，这样可以提高召回率。同时我们也希望用上match\_phrase根据距离提升分数的功能，让几个term距离越近分数就越高，优先返回.就是优先满足召回率，意思，java spark，包含java的也返回，包含spark的也返回，包含java和spark的也返回；同时兼顾精准度，就是包含java和spark，同时java和spark离的越近的doc排在最前面.此时可以用bool组合match query和match\_phrase query一起，来实现上述效果



* 1. 使用rescoring机制优化近似匹配搜索的性能

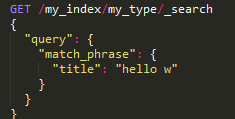


* 1. 通过ngram分词机制实现index-time搜索推荐
     1. ngram

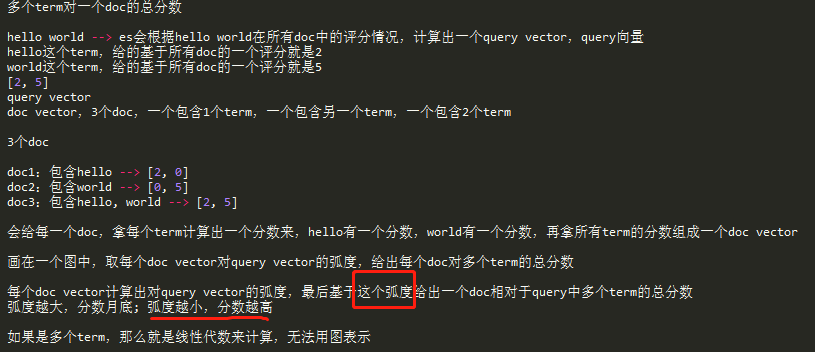
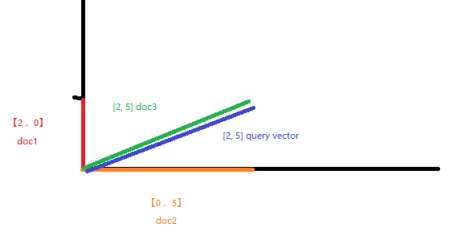


* + 1. 实现index-time搜索推荐

直接在建立索引的时候就建立ngram，这样搜索的时候，不用再根据一个前缀，然后扫描整个倒排索引了; 简单的拿前缀去倒排索引中匹配即可，如果匹配上了，那么就好了; 用match效率更高.

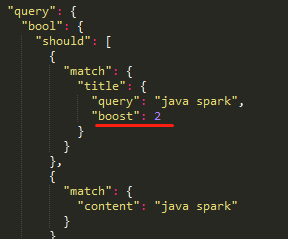


1. 向量空间模型算法(vector space model)



1. 四种常见的相关度分数优化方法
   1. query-time boost

可以增加权重，用的比较多



* 1. 重构查询结构

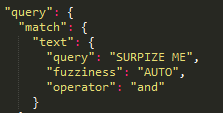
在es新版本中，影响越来越小了。一般情况下，没什么必要的话，大家不用也行

* 1. negative boost
  2. constant\_score

如果你压根儿不需要相关度评分，直接走constant\_score加filter，所有的doc分数都是1，没有评分的概念了

1. fuzzy

fuzzy搜索技术 --> 自动将拼写错误的搜索文本，进行纠正，纠正以后去尝试匹配索引中的数据

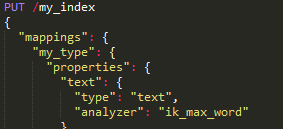


1. IK中文分词
   1. 安装

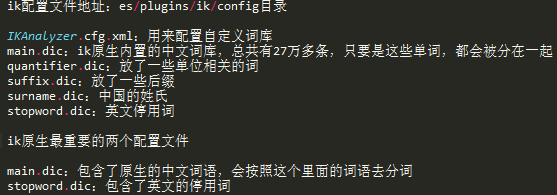
把分词器PLUGIN放到ES/plugin/ik文件夹下，如没有ik文件夹，需要手动建立

* 1. 使用

创建mapping的时候手动指定analyzer,后面就可以用中文分词器来查询了



* 1. 自定义配置



* 1. 修改IK分词器源码来基于mysql热更新词库
     1. 下载源码
     2. 修改ik分词器源码，然后手动支持从mysql中每隔一定时间，自动加载新的词库
     3. mvn package打包代码
     4. 解压缩ik压缩包，将mysql驱动jar，放入ik的目录下
     5. 重启es