1. **前缀搜索、通配符搜索、正则搜索 fuzzy**
   1. 以xx开头的搜索，不计算相关度评分，和filter比，没有bitcache。前缀搜索，尽量把前缀长度设置的更长，性能差。
   2. 语法
      1. 前缀

GET index/\_search

{

"query": {

"prefix": {

"title": {

"value": "text"

}

}

}

}

index\_prefixes: 默认 "min\_chars" : 2, "max\_chars" : 5

* + 1. 通配符：通配符运算符是匹配一个或多个字符的占位符。例如，\*通配符运算符匹配零个或多个字符。您可以将通配符运算符与其他字符结合使用以创建通配符模式

GET my\_index/\_search

{

"query": {

"wildcard": {

"title": {

"value": ""

}

}

}

}

* + 1. 正则：  
       regexp查询的性能可以根据提供的正则表达式而有所不同。为了提高性能，应避免使用通配符模式，如.\*或 .\*?+未经前缀或后缀

**ALL (Default)**

启用所有可选操作符。

**COMPLEMENT**

启用~操作符。可以使用~对下面最短的模式进行否定。例如

a~bc # matches 'adc' and 'aec' but not 'abc'

**INTERVAL**

启用<>操作符。可以使用<>匹配数值范围。例如

foo<1-100> # matches 'foo1', 'foo2' ... 'foo99', 'foo100'

foo<01-100> # matches 'foo01', 'foo02' ... 'foo99', 'foo100'

**INTERSECTION**

启用&操作符，它充当AND操作符。如果左边和右边的模式都匹配，则匹配成功。例如:

aaa.+&.+bbb # matches 'aaabbb'

**ANYSTRING**

启用@操作符。您可以使用@来匹配任何整个字符串。  
您可以将@操作符与&和~操作符组合起来，创建一个“everything except”逻辑。例如:

@&~(abc.+) # matches everything except terms beginning with 'abc'

1. **Fuzzy模糊查询**

混淆字符 (box → fox) 缺少字符 (black → lack)

多出字符 (sic → sick) 颠倒次序 (act → cat)

* 1. **语法：**

GET /\_search

{

"query": {

"fuzzy": {

"user": {

"value": "keyword"

}

}

}

}

* 1. **参数：**
     1. value：（必需，字符串）
     2. fuzziness：（可选，字符串）最大误差 并非越大越好, 召回率高 但是结果不准确
        1. 两段文本之间的Damerau-Levenshtein距离是使一个字符串与另一个字符串匹配所需的插入、删除、替换和调换的数量
        2. 距离公式：Levenshtein是lucene的，es改进版：Damerau-Levenshtein，

axe=>aex Levenshtein=2 Damerau-Levenshtein=1

* + 1. max\_expansions：可选，整数）匹配的最大词项数量。默认为50。
    2. prefix\_length：创建扩展时保留不变的开始字符数。默认为0

避免在max\_expansions参数中使用较高的值，尤其是当prefix\_length参数值为时0。max\_expansions由于检查的变量数量过多，参数中的高值 可能导致性能不佳。

* + 1. transpositions：（可选，布尔值）指示编辑是否包括两个相邻字符的变位（ab→ba）。默认为true。
    2. rewrite：（可选，字符串）用于重写查询的方法

[https://www.elastic.co/cn/blog/found-fuzzy-search#performance-considerations](https://www.elastic.co/cn/blog/found-fuzzy-search" \l "performance-considerations)

1. **match\_phrase\_prefix：最简陋的Suggest**

**xiaomi nf**

* 1. match\_phrase\_prefix与match\_phrase相同,但是它多了一个特性,就是它允许在文本的最后一个词项(term)上的前缀匹配,如果 是一个单词,比如a,它会匹配文档字段所有以a开头的文档,如果是一个短语,比如 "this is ma" ,他会先在倒排索引中做以ma做前缀搜索,然后在匹配到的doc中做match\_phrase查询,(网上有的说是先match\_phrase,然后再进行前缀搜索, 是不对的)

**参数**

analyzer 指定何种分析器来对该短语进行分词处理

max\_expansions 限制匹配的最大词项

boost 用于设置该查询的权重

slop 允许短语间的词项(term)间隔

slop 参数告诉 match\_phrase 查询词条相隔多远时仍然能将文档视为匹配 什么是相隔多远？ 意思是说为了让查询和文档匹配你需要移动词条多少次？

1. **N-gram：token filter**