关于搜索句法（About Search Syntax）

本站搜索引擎允许用户使用标签数据和别的一些元数据定位上传内容，可用数据如上传者和用户产生的计数。搜索引擎也允许用户联合查询标签和元数据，如此进行特定逻辑组合的搜索，从而实现更精确的筛选。本指南旨在解释单个搜索词的句法和特性，并说明如何将单个搜索词组合成复杂的查询句。

搜索词（Search Terms）

确切的搜索需要一组搜索词，上传内容要满足搜索词规定的准则，才会被搜索引擎返回。

标签搜索的行为（Tag Search Behavior）

使用单个搜索词很简单：直接输入你想要的词即可。默认情况下，该词会匹配已建索引的图像标签及其别名。所以，搜索`pinkie pie`时，如你所想，就会得到所有标有且索引了萍琪派的图片。别名也是有索引的，所以搜索别名`ts`和搜索`twilight sparkle`是一回事。

为方便起见，默认的标签搜索有一些特性。标签搜索是大小写不敏感的。就是说，查询句是否大写是无关紧要的。比如说`pinke pie`和`Pinkie Pie`都会返回相同的结果集。

搜索其它字段（Searching Through Other Fields）

其它一些字段也是有索引的，于是可以使用命名空间约定去搜索它们。即，输入字段名加冒号再加目标值。例如，要搜索宽度为1920的图片，可以使用`width`来构造查询词`width:1920`。若某标签自带命名空间，且和某个字段的命名空间相同，此时要查询它，可以用引号等转义。

数值范围的查询（Numeric Range Queries）

数值字段支持可能值范围的查询。可以在字段名的后面加一个句点，再跟随一个限定符（Qualifier），如此表示希望得到数值字段值大于或小于给定的值（或可包括给定值）的结果。想找到得分大于100的图片，可以搜索`score.gt:100`。如果需要大于*或等于*100的得分，则输入`score.gte:100`。下表列举了搜索引擎支持的限定符。

限定符	含义	举例
`gt`	值大于给定值，但不包括给定值	`score.gt:100`
`gte`	值大于或等于给定值	`score.gte:100`
`lt`	值小于给定值，但不包括给定值	`score.lt:100`
`lte`	值小于或等于给定值	`score.lte:100`

日期/时间范围的查询（Date/Time Range Queries）

本站日期与时间值是ISO 8601标准的一个调整后子集。完整的日期由四位数的年、两位数的月和两位数的日确定——年月日由连字符分隔，也即："YYYY-mm-DD"。类似于ISO 8601规定，只需要越右越精确，且没有空挂的连字符，就可以只指定月，甚至只指定年。这种不完整的日期的语义是整个时期（而不是某月的第一天等）。例如，`2015-04`表示2015年4月这一整个月份。

给定了完整的日期，就可以规定该日内的时间。为此，使用`T`或空格分隔后，以时分秒的顺序添加时间。时分秒均为两位数字，且用冒号分隔，也即："HH:MM:SS"。时是24小时制的。和日期类似，若无空挂的冒号，就可以截断时间来指定整个分钟以至整个小时。`2014-04-20 16`表示2014年4月20日下午4点（UTC）的一整个小时。当日的第一个整分钟可以用`2014-04-20 16:00`表示。

默认情况下，时间遵循国际UTC（"Zulu"）时间。（按ISO 8601标准，这相当于暗含了后缀`Z`。）亦可附一个加号或者减号来指定本地时间的偏移量，加号或减号后是两位数的时和两位数的分（分一般是`00`），时分用冒号分隔，例如，`-04:00`可以偏移到美国东部时间（EDT）。注意，和ISO 8061不同的是，本站的日期和时间都能加偏移量，这样可以保证时期段处于用户感兴趣的范围。例如，`2015-05:00`表示在偏移负五小时（美国东部标准时间）的情况下的2015年。

日期/时间的查询也可以使用范围限定符。用`gt`和`lt`不包含指定的时间范围，而用`gte`和`lte`则包含指定的时间范围。

下面举了有效时间查询词的例子。

示例	解释
`created_at:2015`	返回2015年（UTC）的所有上传内容。
`created_at:2015+08:00`	返回2015年（SGT）的所有上传内容。
`created_at:2015-04`	返回2015年4月（UTC）的所有上传内容。
`created_at:2015-04-03:00`	返回2015年4月（BRT）的所有上传内容。
`created_at:2015-04-01`	返回2015年4月1日（UTC）的所有上传内容。
`created_at:2015-04-01+08:00`	返回2015年4月1日（SGT）的所有上传内容。
`created_at:2015-04-01 01`	返回2015年4月1日凌晨1点（UTC）的所有上传内容。
`created_at:2015-04-01 01Z`	返回2015年4月1日凌晨1点（UTC）的所有上传内容。显式指定了UTC（"Zulu"）的0偏移量。
`created_at:2015-04-01T01Z`	返回2015年4月1日凌晨1点（UTC）的所有上传内容。使用了ISO 8601提到的"T"分隔符。
`created_at:2015-04-01 01-04:00`	返回2015年4月1日凌晨1点（EDS）的所有上传内容。
`created_at:2015-04-01 01:00`	返回2015年4月1日凌晨1点1分内（EDS）的所有上传内容。
`created_at:2015-04-01 01:00Z`	返回2015年4月1日凌晨1点1分内（UTC）的所有上传内容。显式指定了UTC（"Zulu"）的0偏移量。
`created_at:2015-04-01 00:00:00`	返回2015年4月1日0点准时（UTC）的所有上传内容。
`created_at:2015-04-01 00:00:00+08:00`	返回2015年4月1日0点准时（SGT）的所有上传内容。
`created_at.lt:2015`	返回2015年（SGT）开始前的所有上传内容。

示例	解释
`created_at.gte:2015-04-04`	返回2015年4月4日（UTC，第5季首播）及之后的所有上传内容。

支持的字段（Supported Fields）

下表列举了搜索引擎支持的所有字段，附有示例。

字段选择符（Field Selector）	类型	描述	示例
`aspect_ratio`	数值范围（Numeric Range）	匹配具有指定长宽比的图片。	`aspect_ratio:1`
`comment_count`	数值范围	匹配具有指定评论数量的图片。	`comment_count.gt:50`
`created_at`	日期/时间范围（Date/Time Range）	匹配在指定日期和/或时间发布的图片。	`created_at:2015-04-01`
`description`	全文段（Full Text）	对具有指定字符串描述的图片进行全文段搜索。	`description:derp`
`downvotes`	数值范围	匹配具有指定踩票数量的图片。	`downvotes:0`
`faved_by`	字面量（Literal）	匹配指定用户收藏的图片。大小写不敏感。	`faved_by:roboshi`
`faves`	数值范围	匹配具有指定收藏数量的图片。	`faves:20`
`height`	数值范围	匹配具有指定高度的图片。	`height:1080`
`id`	数值范围	匹配具有指定编号的图片。	`id:111111`

字段选择符（Field Selector）	类型	描述	示例
`orig_sha512_hash`	字面量	匹配*原始*图片SHA-512校验和。	`orig_sha512_hash:880da*`
`score`	数值范围	匹配具有指定净得分的图片。	`score.gt:200`
`sha512_hash`	字面量	匹配图片SHA-512校验和。注意：图像优化通常会改变其原始校验和。	`sha512_hash:880da*`
`source_url`	字面量	匹配图片的源URL。大小写不敏感。	`source_url:*deviantart.com*`
`tag_count`	数值范围	匹配具有指定标签数的图片。	`tag_count.gt:10`
`uploader`	字面量	匹配指定上传者的图片。大小写不敏感。	`uploader:k_a`
`upvotes`	数值范围	匹配具有指定赞票数量的图片。	`upvotes.gt:200`
`width`	数值范围	匹配具有指定宽度的图片。	`width:1920`

字段选择符（Field Selector）	类型	描述	示例
`wilson_score`	数值范围	匹配在99.5%威尔逊置信区间（Wilson CI）具有指定区间下限的图片。	`wilson_score.gt:0.9`

注意，某些字段没被列出，比如：`artist`和`spoiler`。它们属于*标签命名空间（tag namespaces）*，而非元数据，但功能用法是相同的。所以，搜索`spoiler:s04`可以得到预期结果。

特殊字符和后缀（Special Characters and Suffixes）

通配符（Wildcards）

通配符可以用来匹配特定开头、结尾、或者包含给定字符串的词。类似于文件管理器的通配符，搜索引擎可识别两种通配符：星号和问号。

星号可以“展开到”，或者说，可以匹配到任意数量的字符，包括空字符。例如：`apple*` 可匹配有`apple bloom`，`applejack`以及`apple`等标签的上传内容。

问号则是定位地匹配单个字符。例如，`t?ixie`可以匹配`trixie`或`twixie`。

通配符	匹配
`*`	任意个字符，包括零个
`?`	单个字符

特殊字符的转义（Escaping Special Characters）

要使用会修正搜索词的，或在搜索词之外的特殊字符，需要进行“转义”，以免其跳出了搜索词。要在搜索词中使用特殊字符，需要使用两种常规的字符串转义机制：加反斜杠和加引号。以下是可能需要转义的特殊字符和字符序列：

- `(`
- `)`
- `*`
- `?`
- `-`（当其位于一个搜索词前时）
- `!`（当其位于一个搜索词前时）
- `,`
- `&&`

- `||`
- `OR` （若全大写）
- `AND` （若全大写）
- `NOT` （若全大写）
- `"`
- `\`
- `~` （使用模糊语法匹配时）
- `^` （使用提升词时）

反斜杠可以放在一个特殊字符前（或是上表内的特殊序列前），它能迫使给定字符算作其前或其后搜索词的一部分。如果反斜杠放在一般字符前，则实际上不会有任何作用。例如，`\-_-`可以强制搜索表情符号`-_-`，如果不加反斜杠则遵循了`否定`语法。再来看一个：`rose \(flower\)`——虽说括号有直判规则，一般不需要用转义符。反斜杠本身也是特殊字符，必须转义；字面的反斜杠可以用`\\`表示。

还有一种转义方式是直接用双引号（`"`）包裹搜索词，例如：`"rose (flower)"`。如果要搜索特定字段，双引号**必须包裹字段名和冒号**，例如：`"width:1920"`。在双引号中的内容均会被认作字面，仅有一个例外。注意，双引号字符本身限定了搜索词的边界，所以如果双引号出现在了搜索词内，就必须用反斜杠转义。**此外，双引号内的反斜杠均按字面处理。**

字符串近似（模糊）匹配（Approximate (Fuzzy) String Matching）

本站搜索引擎后端是Apache Lucene，支持所谓字符串“模糊”匹配。字符串模糊匹配可搭配任意搜索词使用，包括默认标签字段。模糊匹配需要一个从0到1.0的相似度量值或一个整数。整数表示*字符串校正的最优编辑距离（optimal string alignment edit distance）*，规定了让一字符串等于给定字符串所必需的最多的编辑次数，编辑可以是删除、插入、替换以及交换相邻字符。指定0到1.0的相似系数亦可，1.0最不“模糊”。相似系数可以派生出一个编辑距离：搜索词长度（不考虑字段名）乘以单位差量，减去相似系数，向下取整。要使用模糊匹配，需要在搜索词后添加波浪线，再添加编辑距离或相似系数。**注意，两种写法都会被Lucene优化封顶到2的编辑距离。所以，过大的编辑距离或过小的相似系数并不会如其字意地返回结果。**

例如，`fluttersho~0.8`会匹配存在和`fluttersho`近似的标签的上传内容，相似系数是0.8。编辑距离是 |(1 − 0.8)(10)| = 2。注意到`fluttershy`是在返回结果中的。这个功能的作用很显然：如果你不能准确拼写某个字符或标签，这就帮得上你的忙。模糊匹配类似于谷歌著名的（或者说，臭名昭著的）拼写检查的手动版。

数值查询也可以添加模糊性以指定范围。这种情况下，模糊参数表示给定数字的偏离量。例如，`width:800~200`确定的图片宽度是从600（800 − 200）到1000（800 + 200）的，包括边界。

模糊匹配可以在表达式的任一项里随意使用。

搜索语法：词算符及其组合（Search Grammar: Term Operators and Combinations）

表达式（Expressions）

搜索词的组合定义查询句，查询句确定结果集。搜索词的组合形式化表示为**表达式**，表达式由搜索词、算符甚至别的表达式组成。组成别的表达式的表达式被称为**子表达式（subexpressions）**。搜索前端可识别的表达式包括词或子表达式的否定，词和子表达式的取式，或两词、两子表达式的取式。

搜索表达式本质上是**二元（binary）**或**一元（unary）**的。二元表达式由一个搜索词或子表达式、一个**算符（operator）**和另一个搜索词或子表达式组成。二元表达式可以由算符“链接”上又一个搜索词。一元表达式由一个算符及跟随算符的单个搜索词或子表达式组成。下面几节介绍表达式类型和子表达式使用方式。

总表（Summary Table）

算符	符号	注释
否定（非）	`NOT`；`-`；`!`	用于单词或括起的子表达式前。无需向右填充。表示*不要匹配*对应搜索词或子表达式。
合取（与）	`,`；`&&`；`AND`	用于两词之间。逗号两端空格填充是可选的；别的形式必须填充空格。表示同时匹配两个词。可以继续链接更多词。
析取（或）	`||`；`OR`	用于两词之间，必须包裹空格。表示匹配任一词。可以继续链接更多词。

否定（Negation）

一个词或表达式的**否定**表示原始词或子表达式*不应被*匹配。否定算符是**一元**的，也即用于单个词或单个子表达式。可以用全大写的`NOT`表示，用非全角变体的分字符`-`表示，或者使用叹号`!`表示。例如：`-fluttershy`或`NOT fluttershy`匹配*没有*`fluttershy`标签的图片。用集合论术语来说，这是补集操作，即上传全集在此之外的内容。

逗号和与表达式（Commas and AND Expressions）

查询满足*所有*词的表达式被称为**合取式（conjunction）**或**与表达式**。和过去一样，可以用逗号链接搜索词以查询符合一系列条件的图片。例如，`fluttershy,pinkie pie`返回*同时有*`fluttershy`*和*`pinkie pie`标签的图片。用集合论术语来说，该结果集是带`fluttershy`标签的上传集和带`pinkie pie`标签的上传集的交集。

逗号的空格填充是任意的。和过去不一样，逗号现在就是与算符，因此更加通用。逗号可以在子表达式中使用，也可以和或算符并用，下文将详细讨论。

与算符也能用`&&`（源自典型的编程记号）或是全大写的`AND`来表示，例如：`rarity && pinkie pie`或`rarity AND pinkie pie`。和逗号不一样，这两种表示需要在算符两边填充空格。

或表达式（OR Expressions）

析取式（disjunction）或称**或表达式**请求的是满足*任一*搜索词的上传内容。它明显不同于前述的与表达式。重申一下，与表达式要求结果满足*所有*词。或算符可以用`||`（也是一种编程记号）或全大写的`OR`表示，例如：`rarity || pinkie pie`或`rarity OR pinkie pie`。用集合论术语来说，该结果集是带`rarity`标签的上传集和带`pinkie pie`标签的上传集的并集。所有的或算符都需要在两边填充空格。

复合表达式（Compound Expressions）

搜索词的复杂组合引出复杂的搜索准则，可以由表达式的组合来实现。这实际上类似于算数。考虑乘法和加法（在所称的*布尔代数（Boolean Algebra）*中类似于与和或算符）。可以用乘法和加法表示代数式，比如有A，B，C三项，代数式为$A \times B + C$。乘法是要优先于加法的，所以这个表达式等价于$(A \times B) + C$，后者显式说明了运算顺序。

算符优先级（Operator Precedence）

同样地，链式与或非运算可以规定优先级以确定运算顺序。搜索句法中的运算顺序如下：

1. 否定（非）
2. 合取（与）
3. 析取（或）

考虑这样一个查询：`twilight sparkle || fluttershy && pinkie pie`。在这个例子里，`fluttershy && pinkie pie`要优先结算，于是变成一个隐含的*子表达式*。然后，子表达式的结果和`twilight sparkle`做或运算。于是，该搜索句指示搜索引擎返回标有`twilight sparkle`*或*同时标有`fluttershy`*和* `pinkie pie`的上传内容。注意，倘若优先结算`twilight sparkle || fluttershy`，那结果就有所不同了。

用括号规定子表达式（Defining Subexpressions with Parentheses）

回到之前的算术运算的例子，我们可以用显式的子表达式来规定运算的顺序。这种操作需要用到*分隔符（Delimiter）*，它将作为子表达式的边界，而通常是括号承担此任务。于是，$A \times (B + C)$强制$B + C$优先结算，然后结果和A相乘，这和一般顺序相反。同样地，`(twilight sparkle || fluttershy) && pinkie pie`指示搜索引擎返回标有`twilight sparkle`*或*`fluttershy`并且*总是标有*`pinkie pie`的上传内容。

如前所述，一元的非算符可以用于带括号的子表达式。其语义类似于将其用于单个搜索词：被否定的子表达式确定了*不*遵从原子表达式规定的上传内容。例如：查询`-(pinkamena diane pie, grimdark)`，返回所有*并未*同时标有`pinkamena diane pie`*和*`grimdark`的上传内容。有其中一项的上传内容还是会被返回，只要不是两者兼具。因此，轻松愉快的萍卡美娜图片以及没有萍卡美娜的黑暗图片将出现，但交集（萍卡美娜的黑暗图片）将被排除在外。

带括号的显式子表示允许复杂查询，这样的子表达式还能进一步嵌套在其它子表达式中以进一步微调结果集。

自动括号转义（Automatic Parentheses Escaping）

最后，关于括号还是值得备注一下的。一般的话，如果表达式解析器遇到了没有闭合的开括号，则会抛出错误。本搜索引擎就是这么个情况，而且也会在解析错误页面突出提示这事。不过，在一定限制下，搜索词内部可以有括号——只要词内括号是闭合的，而且并不括起整个词。第一个限制是启发式的，和括号的惯常用法相合；第二个限制是因为括号能括出单个搜索词。因此，搜索`rose (flower)`就会搜出带`rose (flower)`标签的上传内容；而搜索表情符号`))B-(`则会抛出错误；同时，`(q)`实际上等价于`q`。遇到后两种情况时，可以用双引号包裹搜索词，好让搜索引擎明白你的意思。

提升词（Boosting Terms）

搜索引擎还允许在搜索结果按相关性（Relevance）排序时使用提升操作，这样可以让包括或者不包括给定搜索词的上传内容优先或者滞后显示。提升的原理是调整一个搜索词的相关性分数，或增或减。调整量放在搜索词之后，插入符（`^`）之前，它是一个或正或负的十进制数字。例如：`pinkie pie^1 || tara strong`返回标有`pinkie pie`或`tara strong`的上传内容，而当以相关性排序时，标有`pinkie pie`的内容是优先的。同时，负值会降低相关性分数，并会在以相关性排序时降低优先级，例如：`pinkie pie^-1 || tara strong`。排序选项见于搜索框之下，要想让提升词正确起作用，则**必须将排序设置为按相关性排序**。于是，在这两种情况下，*同时*有两个标签的图片依然会优先出现。

附注

译文说明

本译文仅供参考，原文见https://trixiebooru.org/pages/search_syntax
本译文修改了原文超链接，使之符合字面，并增加了两个缺少的示例
本译文发表于2024-01-06

原文纠错

虽然原指南在2023年10月有过更新，但实际上仍然和搜索引擎实际表现脱节，且指南并未说明一些重要的非tag namespaces字段（如`faved_by_id`和`my`），但补充这些内容并非翻译的任务，读者可以自行探索额外内容，以下仅指出译者发现的原文示例中的错误。

1. 在日期/时间范围的查询一节中，示例提到`created_at:2015-04-03:00`指示的时间段相当于按西三区时间来算的2015年4月，然而实际查询却出现了错误。同时，`created_at:2015-04-01+08:00`使用加号，是正常返回的，推测是解析器将小时偏移的负号解释成了一个空挂的分词符。
2. 在字符串近似匹配一节中，提到了可以使用范围为0到1的相似系数，但实际查询时，似乎均按编辑距离来解释了，也就是说示例中的`fluttersho~0.8`不能得到期望结果，应改为`fluttersho~2`。
3. 在提升词一节中，boost value实际上不能是负数。