英　　相　而言比　容易　　：　　之　都是以空格或者（一些）　点隔　。　　然而即使在英　　中也会有一些争　：*you're* 是一个　　是　个？*o'clock*，*cooperate*，*half-baked*，或者 *eyewitness*　些　？

　或者荷　　把独立的　　合并起来　造一个　的合成　如　*Weißkopfseeadler*　(white-headed　sea eagle)，但是　了在　　`Adler`　(eagle)的　候返回　　`Weißkopfseeadler`　的　果，我　需要　得　将合并　拆成　　。

　洲的　言更　　：很多　言在　　，句子，甚至段落之　没有空格。　　有些　可以用一个字来表，但是同　的字在　一个字旁　的　候就是不同意思的　　的一部分。

　而易　的是没有能　奇　般　理所有人　　言的万能分析器，Elasticsearch　　很多　言提供了用的分析器，其他特殊　言的分析器以　件的形式提供。

然而并不是所有　言都有　用分析器，而且有　候　甚至无法　定　理的是什　　言。　　情况，我　需要一些忽略　言也能合理工作的　准工具包。

准分析器

任何全文　索的字符串域都　　使用　`standard`　分析器。　　如果我　想要一个　自定　　分析器，可以按照如下定　方式重新　　准 分析器：

```
{
    "type":      "custom",
    "tokenizer": "standard",
    "filter":  [ "lowercase", "stop" ]
}
```

在 [token-normalization]（　准化　　元）和 [stopwords]（停用　）中，我　　了 `lowercase`（小写字母）和 `stop`（停用　）　　元　　器，但是　在，我　　注于 `standard` *tokenizer*（　准分　器）。

准分　器

分　器　接受一个字符串作　　入，将　个字符串拆分成独立的　或　　元（*token*）（可能会　　一些　点符号等字符），然后　出一个　　元流（*token stream*）。

有趣的是用于　　的算法。　`whitespace`　（空白字符）分　器按空白字符　——　空格、tabs、　行符等等　行　　拆分 —— 然后假定　　的非空格字符　成了一个　　元。例如：

```
GET /_analyze?tokenizer=whitespace
You're the 1st runner home!
```

　个　求会返回如下　　（terms）：`You're`、`the`、`1st`、`runner`、`home!`

letter 分 器，采用 外一 策略，按照任何非字符 行拆分， 将会返回如下 ：You 、 re 、 the 、 st 、 runner 、 home 。

standard 分 器使用 Unicode 文本分割算法 （定 来源于 Unicode Standard Annex #29）来 之 的界限，并且 出所有界限之 的内容。 Unicode 内含的知 使其可以成功的 包含混合 言的文本 行分 。

点符号可能是 的一部分，也可能不是， 取决于它出 的位置：

```
GET /_analyze?tokenizer=standard
You're my 'favorite'.
```

在 个例子中，You're 中的 号被 的一部分，然而 'favorite' 中的 引号 不会被 的一部分， 所以分 果如下：You're 、 my 、 favorite 。

TIP

uax_url_email 分 器和 standard 分 器工作方式 其相同。 区 只在于它能 email 地址和 URLs 并 出 个 元。 standard 分 器 不一 ，会将 email 地址和 URLs 拆分成独立的 。例如，email 地址 joe-bloggs@foo-bar.com 的分 果 joe 、 bloggs 、 foo 、 bar.com 。

standard 分 器是大多数 言分 的一个合理的起点，特 是西方 言。 事 上，它 成了大多数特定 言分析器的基 ，如 english 、french 和 spanish 分析器。 它也支持 洲 言，只是有些 陷，可以考 通 ICU 件的方式使用 icu_tokenizer 行替 。

安装 ICU 件

Elasticsearch的 ICU 分析器 件 使用 国 化 件 *Unicode* (ICU) 函数 （ 情 看 site.project.org ）提供 富的 理 Unicode 工具。 些包含 理 洲 言特 有用的 icu_分 器 ， 有大量 除英 外其他 言 行正 匹配和排序所必 的分 器。

NOTE

ICU 件是 理英 之外 言的必需工具，非常推 安装并使用它，不幸的是，因 是基于 外的 ICU 函数 ， 不同版本的ICU 件可能并不兼容之前的版本，当更新 件的 候， 需要重新索引 的数据。

安装 个 件，第一 先 掉 的Elasticsearch 点，然后在Elasticsearch的主目 行以下命令：

```
./bin/plugin -install elasticsearch/elasticsearch-analysis-icu/$VERSION ①
```

① 当前 $VERSION （版本）可以在以下地址 到 *https://github.com/elasticsearch/elasticsearch-analysis-icu*.

一旦安装后，重 Elasticsearch， 将会看到 似如下的一条 日志：

```
[INFO][plugins] [Mysterio] loaded [marvel, analysis-icu], sites [marvel]
```

如果 有很多 点并以集群方式 行的， 需要在集群的 个 点都安装 个 件。

icu_分　器

icu_分　器 和　准分　器 使用同　的 Unicode 文本分段算法，　只是　了更好的支持　洲　，添加了泰　、老　　、中文、日文、和　文基于　典的　　方法，并且可以使用自定　　将　　和柬埔　文本拆分成音　。

例如，分　比　　准分　器 和 icu_分　器 在分　泰　中的 'Hello.　I　am　from　Bangkok.'　生的　　元：

```
GET /_analyze?tokenizer=standard
```

准分　器　生了　个　　元，　个句子一个：　　，　　。　个只是　想搜索整个句子 'I am from Bangkok.' 的　候有用，但是如果　　想搜索 'Bangkok.'　不行。

```
GET /_analyze?tokenizer=icu_tokenizer
```

相反，　icu_分　器　可以把文本分成独立的　　（　　，　　，　　，　　，　　），　使得文　更容易被搜索到。

相　而言，　　准分　器　　分　中文和日文的　候“　度分　”了，　常将一个完整的　拆分　独立的字符，因　　之　并没有空格，很　区分　　的字符是　隔的　　是一个句子中的　字：

- 向的意思是　*facing*　（面　），　日的意思是　*sun*　（太　），葵的意思是　*hollyhock*（蜀葵）。当写在一起的　候，向日葵的意思是 *sunflower*（向日葵）。
- 五的意思是 *five*（五）或者 *fifth*（第五），　月的意思是 *month*（月　），雨的意思是 *rain*（下雨）。　第一个和第二个字符写在一起成了五月，意思是 *the month of May*（一年中的五月），然而添加上第三个字符，　　五月雨的意思是　　*continuous　　rain*　　（　　不断的下雨,梅雨）。当在合并第四个字符，　式，　意思是 *style*（　式），五月雨式　个　　成了一　不屈不　持　不断的　西的形容　。

然　个字符本身可以是一个　　，但使　　元保持更大的原始概念比使其　作　一个　　的一部分要有意　的多：

```
GET /_analyze?tokenizer=standard
向日葵

GET /_analyze?tokenizer=icu_tokenizer
向日葵
```

准分　器 在前面的例子中将　个字符　出　　独的　　元：向，日，葵。 icu_分　器 会　出个　　元 向日葵（sunflower）。

<code>　准分　器</code>　　和　　<code>icu_分　器</code>　　的　一个不同的地方是后者会将不同　写方式的字符（例如，<code>βeta</code>　）拆分成独立的　　元 — <code>β</code> 和

<code>eta</code>—，而前者 会 出 个 元：<code>βeta</code>。

整理 入文本

当 入文本是干 的 候分 器提供最佳分 果，有效文本， 里 有效 指的是遵从 Unicode 算法期望的 点符号 。 然而很多 候，我 需要 理的文本会是除了干 文本之外的任何文本。在分 之前整理文本会提升 出 果的 量。

HTML 分

将 HTML 通 准分 器 或 icu_分 器 分 将 生糟 的 果。 些分 器不知道如何 理 HTML 。例如：

```
GET /_analyze?tokenizer=standard
<p>Some d&eacute;j&agrave; vu <a href="http://somedomain.com">website</a>
```

准分 器会混 HTML 和 体，并且 出以下 元：p、Some、d、eacute、j、agrave、vu、a、href、http、somedomain.com、website、a。 些 元 然不知所云！

字符 器 可以添加 分析器中，在将文本 分 器之前 理 文本。在 情况下，我 可以用 <code>html_strip</code> 字符 器移除 HTML 并 HTML 体如 <code>é</code> 一致的 Unicode 字符。

字符 器可以通 analyze API 行 ， 需要在 字符串中指明它 ：

```
GET /_analyze?tokenizer=standard&char_filters=html_strip
<p>Some d&eacute;j&agrave; vu <a href="http://somedomain.com">website</a>
```

想将它 作 分析器的一部分使用，需要把它 添加到 custom 型的自定 分析器里：

```
PUT /my_index
{
    "settings": {
        "analysis": {
            "analyzer": {
                "my_html_analyzer": {
                    "tokenizer":     "standard",
                    "char_filter": [ "html_strip" ]
                }
            }
        }
    }
}
```

一旦自定 分析器 建好之后， 我 新的 my_html_analyzer 就可以用 analyze API ：

```
GET /my_index/_analyze?analyzer=my_html_analyzer
<p>Some d&eacute;j&agrave; vu <a href="http://somedomain.com>">website</a>
```

次 出的 元才是我 期望的： Some， déjà， vu， website 。

整理 点符号

准分 器 和 icu_分 器 都能理解 中的 号 当被 的一部分，然而包 的引号在不 。分 文本 You're my 'favorite'， 会被 出正 的 元 You're ， my ， favorite 。

不幸的是， Unicode 列出了一些有 会被用 号的字符：

U+0027
号 (<code>'</code>)— 原始 ASCII 符号

U+2018
左 引号 (<code>‘</code>)— 当 引用 作 一个引用的 始

U+2019
右 引号 (<code>’</code>)— 当 引用 座位一个引用的 束，也是 号的首 字符。

当 三个字符出 在 中 的 候， 准分 器 和 icu_分 器 都会将 三个字符 号（会被 的一部分）。然而 有 外三个 得很像 号的字符：

U+201B
Single high-reversed-9 （高反 引号） (<code>‛</code>)— 跟 <code>U+2018</code> 一 ，但是外 上有区

U+0091
ISO-8859-1 中的左 引号 — 不会被用于 Unicode 中

U+0092
ISO-8859-1 中的右 引号 — 不会被用于 Unicode 中

准分 器 和 icu_分 器 把 三个字符 的分界 — 一个将文本拆分 元的位置。不幸的是，一些出版社用 U+201B 作 名字的典型 写方式例如 M‛coy ， 第二个字符或 可以被 的文字 理 件打出来， 取决于 款 件的年 。

即使在使用可以“接受”的引号 ，一个用 引号 写的 — <code>You're</code> — 也和一个用 号 写的 — <code>You’re</code> — 不一 ，意味着搜索其中的一个 体将会 不到 一个。

幸 的是，可以用 mapping 些混乱的字符 行分 ， 器可以 行我 用 一个字符替 所有 例中的一个字符。 情况下，我 可以 的用 U+0027 替 所有的 号 体：

```
PUT /my_index
{
  "settings": {
    "analysis": {
      "char_filter": { ①
        "quotes": {
          "type": "mapping",
          "mappings": [ ②
            "\\u0091=>\\u0027",
            "\\u0092=>\\u0027",
            "\\u2018=>\\u0027",
            "\\u2019=>\\u0027",
            "\\u201B=>\\u0027"
          ]
        }
      },
      "analyzer": {
        "quotes_analyzer": {
          "tokenizer":     "standard",
          "char_filter": [ "quotes" ] ③
        }
      }
    }
  }
}
```

① 我 自定 了一个 char_filter （字符 器）叫做 quotes，提供所有 号 体到 号的映射。

② 了更清晰，我 使用 个字符的 JSON Unicode 句，当然我 也可以使用他 本身字符表示："‘⇒'"。

③ 我 用自定 的 quotes 字符 器 建一个新的分析器叫做 quotes_analyzer。

像以前一 ，我 需要在 建了分析器后 它：

```
GET /my_index/_analyze?analyzer=quotes_analyzer
You’re my ‘favorite’ M‘Coy
```

个例子返回如下 元，其中所有的 中的引号 都被替 了 号：You’re, my, favorite, M‘Coy。

投入更多的努力 保 的分 器接收到高 量的 入， 的搜索 果 量也将会更好。