Computer version

NLP_ Information-seeking Tasks

Text Classification

Text Classification是NLP中一个重要的任务，其目标是将一段文本分配到预定义的类别中。例如：

- **语言识别**：判断一段文本是英语、中文还是其他语言。
- **流派分类**：将一本书或电影的描述归为科幻、喜剧等类别。
- **垃圾邮件检测**：判断一封邮件是“垃圾邮件（spam）”还是“正常邮件（ham）”。

示例：

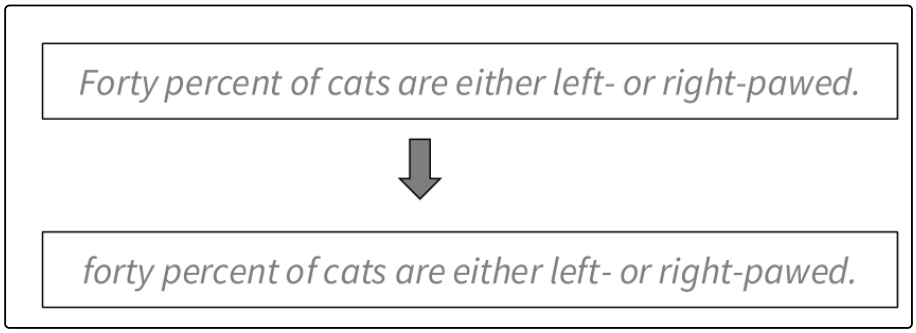
- 垃圾邮件（Spam）的特点：
  - 包含大量广告词汇，如“buy now”，“discount”。
  - 通常会使用不规范的字符混淆，例如“ViagraFr$1.85”。
- 正常邮件（Ham）的特点：
  - 语言自然，没有明显的广告倾向。

你可以将垃圾邮件分类类比为一个超市工作人员检查货物是否损坏。垃圾邮件类似有瑕疵的货物，正常邮件则是可以直接上架的商品。

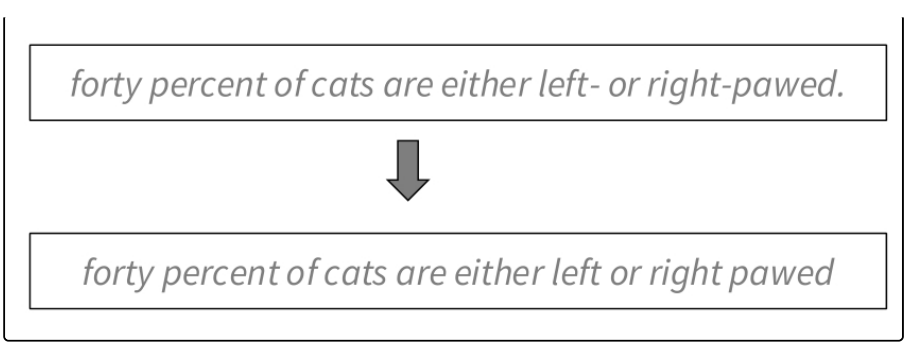
Pre-processing Text

在对文本进行分类或其他NLP任务之前，我们需要对文本进行预处理。这就像在厨房做饭前先准备食材。以下是常见的预处理步骤：

1. Case-folding：
   - 将所有字符转为小写。例如，将“HELLO World”转换为“hello world”。这样可以避免大小写的差异影响分析。

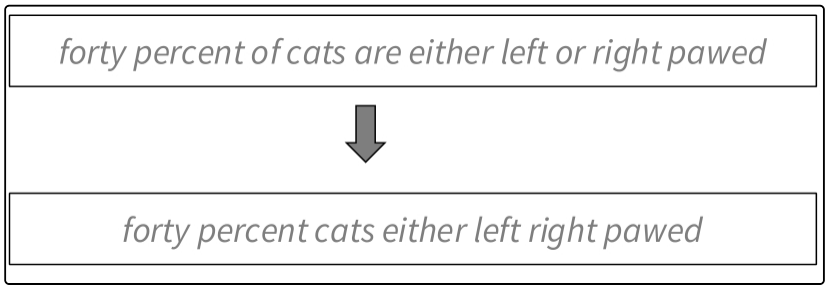


2. Removal of Punctuation：
   - 移除标点符号，如“.”、“,”。例如，“Hello, world!”变为“Hello world”。



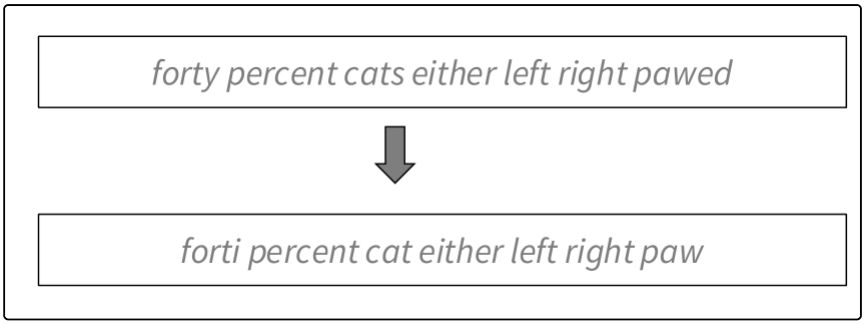
3. **Removal of Stop Words**：

- 去除常见但无实际意义的词，如“the”、“is”、“and”。这些词类似于“背景噪声”，去掉后有助于突出文本的核心信息。

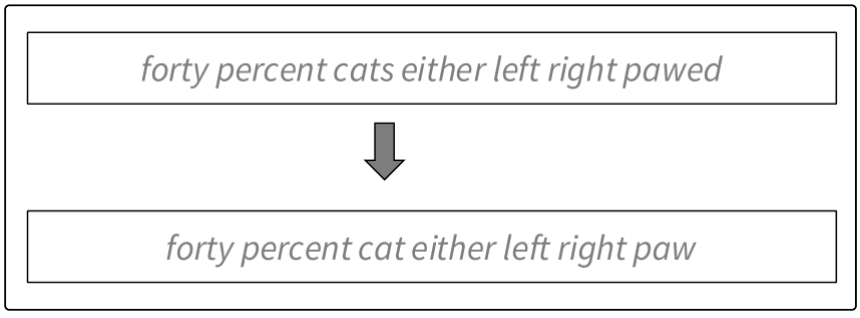


4. **Normalisation**：

- **Stemming**：将单词还原为词干。例如，“running”和“runner”还原为“run”。



- **Lemmatization**：将单词还原为标准形式。例如，“better”还原为“good”。



5. **Tokenisation**：

- 将文本分解为单词或短语。例如，“The cat sleeps”被分为[“The”，“cat”，“sleeps”]。

🌐 垃圾邮件检测

假设你有以下两封邮件：

- **邮件 1**：Buy cheap watches now!
- **邮件 2**：Hey, how was your day?

预处理步骤：

1. **Case-folding**：

- 邮件 1：buy cheap watches now!
- 邮件 2：hey, how was your day?

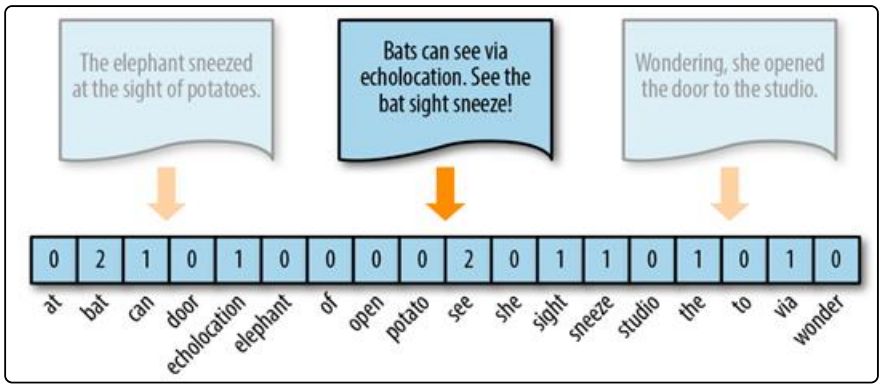
2. **Removal of Punctuation**：
   - 邮件 1：buy cheap watches now
   - 邮件 2：hey how was your day
3. **Removal of Stop Words**：
   - 邮件 1：buy cheap watches
   - 邮件 2：hey day
4. **Normalisation**：
   - 邮件 1 和邮件 2：无明显变化。

经过这些步骤，我们可以更高效地提取出有用的信息，从而进行分类。

Text Classification: Bag of Words (BoW)

概念解释：

Bag of Words（BoW）是一种经典的文本表示方法。其核心思想是将文本表示为“词袋”，即将每个单词视为一个特征，并记录这些单词在文本中出现的次数。



表示方式：

- **Features**：文本中的单词（unigram tokens）。
- **Values**：每个单词在文本中出现的次数（词频）。

举例说明：

假设我们有两段文本：

1. “The elephant sneezed at the sight of potatoes.”
2. “Bats can see via echolocation. See the bat sight sneeze!”

步骤：

1. **词汇提取**：
   - 提取所有唯一单词，例如：["the", "elephant", "sneezed", "sight", "potatoes", "bats", "can", "see", "via", "echolocation", "bat", "sneeze"]。
2. **特征向量生成**：
   - 每段文本对应一个向量，记录每个单词的词频。
   - 示例向量：

- 文本 1：[2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]
- 文本 2：[1, 0, 1, 2, 0, 1, 1, 2, 1, 1, 1, 1]

- **优势**：
  - 简单易实现。
  - 能很好地捕捉单词的整体分布。
- **劣势**：
  - 丢失了单词的顺序信息。
  - 对于长文本，生成的向量往往是稀疏的（大部分值为零）。

特征选择与分类算法

特征选择：

- **去掉停用词**：例如"the"，"is"，"of"，这些词对分类结果贡献较小。
- **选择高频或低频单词**：去掉频繁出现在所有文档中的单词（因为这些单词对区分类别没有帮助）。

常用分类算法：

1. **支持向量机（SVM）**：通过找到最佳超平面将文本分为不同类别。
2. **决策树**：通过构建分裂规则来分类。
3. **逻辑回归**：一种概率模型，用于预测文本属于某一类别的概率。

垃圾邮件分类

- 文档 1（正常邮件）："Hi, how are you? I wanted to check on your progress."
- 文档 2（垃圾邮件）："Buy cheap medicines now!!! Limited offer."

预处理后生成的BoW特征向量可以输入到分类器中，让分类器学习如何将邮件分类为“正常”或“垃圾”。

Information Retrieval (IR)

核心概念

Information Retrieval (IR) 的目标是满足用户的信息需求，也就是根据用户的查询（query，记为 q）找到相关的文档。

IR 系统的组成部分：

1. **Corpus of Documents(语料库)**：
   - 包含所有文档的集合（文档可以是段落、页面或多页）。
2. **Query**：
   - 用户输入的查询，通常是一个词语序列或关键字。

- 查询可以支持布尔操作符，例如：`[artificial AND intelligence]`.

3. **Result Get**:

- 返回与查询相关的文档集.

4. **Presentation of Result Set**:

- 按相关性排序的文档列表或网络形式展示.

示例：当你在Google中搜索“machine learning basics”时：

- Corpus 是互联网上所有网页.
- Query 是“machine learning basics”.
- Result Set 是Google返回的相关页面.
- Presentation 是按点击率或相关性排序的搜索结果.

Boolean Keyword Model

核心思想

布尔关键字模型是最早的 IR 方法之一，它的特点是：

- 每个词在语料库中是一个布尔特征.
  - 如果文档 d 包含某个词，则对应的值为 1；否则为 0.
- 查询是由关键词构成的布尔表达式（例如，`artificial AND intelligence`）.

优势：

- 简单直观.
- 易于实现.

劣势：

- 无法对结果进行相关性排序.
- 查询和结果完全依赖布尔逻辑，灵活性较差.

BM25 Scoring Function

数学表达

BM25 是一种基于打分的 IR 模型，其公式为：

$$BM25(d_j, q_{1:N}) = \sum_{i=1}^{N} IDF(q_i) \cdot \frac{TF(q_i, d_j) \cdot (k+1)}{TF(q_i, d_j) + k \cdot (1 - b + b \cdot \frac{|d_j|}{L})}$$

其中：

- $q_{1:N}$ 是查询中的单词序列.
- d_j 是文档.
- $TF(q_i, d_j)$ 是词频，表示单词 q_i 在文档 d_j 中出现的次数.
- $IDF(q_i)$ 是逆文档频率，衡量单词的重要性.
- $|d_j|$ 是文档 d_j 的长度.

- L 是语料库中所有文档的平均长度.
- k 和 b 是调节参数.

工作流程

1. **计算每个词的权重**：通过词频（TF）和逆文档频率（IDF）衡量.
2. **计算文档与查询的相似度**：为每个文档输出一个分数.
3. **排序并返回结果**：根据分数，从高到低返回相关文档.

实际例子：
假设用户的查询是“deep learning applications”，语料库包含以下文档：

- 文档 1："Deep learning revolutionized artificial intelligence."
- 文档 2："Applications of deep learning are vast."

通过 BM25 模型，可以计算每个文档的相关性分数，并返回排序后的结果.

BM25 Scoring: Factors and Explanation

BM25 是一种广泛用于信息检索的打分函数，它结合了以下三个主要因素：

Term Frequency (TF)

- 描述一个单词在文档中出现的频率.
- **直观解释**：单词在文档中出现越频繁，表明该文档更可能与查询相关.
- **示例**：如果查询为“farming in Kansas”，提到“farming”或“Kansas”的文档会有更高的分数.

Inverse Document Frequency (IDF)

- 用于衡量某个单词的重要性. 其公式为：

$$IDF(q_i) = \log\left(\frac{N}{DF(q_i)}\right)$$

  - N 是文档总数.
  - $DF(q_i)$ 是包含单词 q_i 的文档数.
- **直观解释**：常见词（例如“in”）的 DF 高，IDF 低，表明它对区分文档的重要性较低.

Document Length

- 较长的文档可能提及查询中的每个词，但不一定是核心内容.
- **直观解释**：BM25 对文档长度进行归一化，防止长文档因累积词频而获得更高的分数.

BM25 Scoring Formula

BM25 的计算公式

$$BM25(d_j, q_{1:N}) = \sum_{i=1}^{N} IDF(q_i) \cdot \frac{TF(q_i, d_j) \cdot (k+1)}{TF(q_i, d_j) + k \cdot (1 - b + b \cdot \frac{|d_j|}{L})}$$

其中：

- $q_{1:N}$ 是查询中的词序列。
- d_j 是文档。
- $TF(q_i, d_j)$ 是单词 q_i 在文档 d_j 中的出现频率。
- $|d_j|$ 是文档长度。
- L 是语料库的平均文档长度。
- k 和 b 是调节参数，通常取 $k = 2.0$ 和 $b = 0.75$。

结合示例分析

查询：(farming in Kansas)

假设语料库包含以下文档：

1. 文档 1："Farming is a major industry in Kansas."
2. 文档 2："Kansas City hosts several farming exhibitions."

- 对于"farming"和"Kansas"，它们的 TF 和 IDF 会被分别计算。
- 文档 1 和文档 2 的 BM25 分数会根据公式加总计算：

$$BM25(d_1, [farming, kansas]) = IDF(farming) \cdot \text{TF Component} + IDF(kansas) \cdot$$

分数影响的直观解释：

- 如果文档中某个词出现的频率很高（高 TF），且这个词在其他文档中出现的频率较低（高 IDF），该文档的相关性得分会更高。
- 长文档的分数会被适当调整，以避免长度造成的偏差。

Inverse Document Frequency (IDF)

概念：

IDF 衡量的是一个单词在整个语料库中的重要性。其公式如下：

$$IDF(q_i) = \log\left(\frac{N - DF(q_i) + 0.5}{DF(q_i) + 0.5}\right)$$

其中：

- N 是文档总数。
- $DF(q_i)$ 是包含单词 q_i 的文档数。
- 公式中的 $+0.5$ 是一种平滑处理，防止分母为零。

直观解释：

- **常见单词**（如"the"、"in"）的 DF 高，因此 IDF 低。
- **稀有单词**（如"quantum"）的 DF 低，因此 IDF 高。

示例：
假设查询单词为 "Kansas"，语料库总文档数 $N = 100$，而提到 "Kansas" 的文档数 $DF(\text{"Kansas"}) = 20$。

$$IDF(\text{"Kansas"}) = \log\left(\frac{100 - 20 + 0.5}{20 + 0.5}\right) = \log\left(\frac{80.5}{20.5}\right)$$

Information Retrieval Evaluation

精度 (Precision) 和召回率 (Recall)

1. **Precision**: 结果集中相关文档所占比例。

$$\text{Precision} = \frac{\text{相关文档数}}{\text{结果集中文档总数}}$$

2. **Recall**: 所有相关文档中被检索出来比例。

$$\text{Recall} = \frac{\text{相关文档数}}{\text{语料库中相关文档总数}}$$

3. **F1 Score**: Precision 和 Recall 的调和平均数。

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

示例:
假设结果集中包含 40 篇文档，其中 30 篇相关，10 篇不相关；语料库中实际相关文档为 50 篇。

	In result set	Not in result set
Relevant	30	20
Not relevant	10	40

- Precision:

$$\frac{30}{30 + 10} = 0.75$$

- Recall:

$$\frac{30}{30 + 20} = 0.60$$

- F1 Score:

$$2 \cdot \frac{0.75 \cdot 0.60}{0.75 + 0.60} = 0.67$$

Refinements in IR

Case-Folding

- 将所有单词转为小写：例如，“COUCH” 转为 “couch”。

Normalization

- 词形还原：例如，“couches” 转为 “couch”。
- 注意：在某些情况下可能影响精度（如 “stocking” 转为 “stock”）。

Synonym Recognition

- 将同义词归为一类：例如，“couch” 和 “sofa”。
- 注意：语境中可能产生歧义（如 “Tim Couch” 中的 “Couch” 不指沙发）。

PageRank Algorithm

核心思想

PageRank 是一种评估网页重要性的算法。即使某些文档中“IBM”被多次提及，但我们期望返回的顶级结果是 IBM 的官方网站，而非其他网页。

PageRank 的关键概念：

1. **In-links**：指向某一页面的链接数量。
2. **Out-links**：从某一页面发出的链接数量。
3. **评分传播**：页面通过链接将自己的 PageRank 评分分配给其他页面。

PageRank 公式：

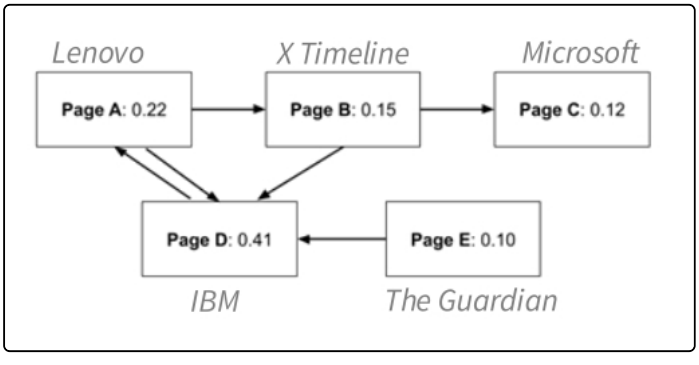
$$PR(p) = \frac{1-d}{N} + d \cdot \sum_{i} \frac{PR(in_i)}{C(in_i)}$$

其中：

- N：总页面数。
- in_i：指向页面 p 的页面。
- $C(in_i)$：页面 in_i 的出链数。
- d：阻尼因子，通常取值 $d = 0.85$，表示用户随机点击链接的概率。

示例计算

假设有 5 个页面：A, B, C, D, E。以下是具体信息：



- 页面 A 的初始 PageRank 为 0.22.
- 页面 B 的初始 PageRank 为 0.15.
- 页面 E 的初始 PageRank 为 0.10.
- 页面 D 接收来自 A, B, E 的链接.

计算页面 D 的 PageRank:

$$PR(D) = \frac{1 - 0.85}{5} + 0.85 \cdot \left(\frac{PR(A)}{C(A)} + \frac{PR(B)}{C(B)} + \frac{PR(E)}{C(E)} \right)$$

其中:

- $C(A) = 2$, 即 A 链接到 2 个页面.
- $C(B) = 2$, 即 B 链接到 2 个页面.
- $C(E) = 1$, 即 E 链接到 1 个页面.

代入数值:

$$PR(D) = \frac{1 - 0.85}{5} + 0.85 \cdot \left(\frac{0.22}{2} + \frac{0.15}{2} + \frac{0.10}{1} \right)$$

$$PR(D) = 0.03 + 0.85 \cdot (0.11 + 0.075 + 0.10)$$

$$PR(D) = 0.03 + 0.85 \cdot 0.285 = 0.03 + 0.24225 = 0.27225$$

页面 D 的最终 PageRank 约为 0.27.

---

1. **链接传递重要性**:
   - 来自高 PageRank 页面的链接比低 PageRank 页面的链接更重要.
   - 页面 E 的链接权重更高, 因为它的出链数更少 ($C(E) = 1$).
2. **阻尼因子 (Damping Factor)**:
   - 用于模拟用户随机跳转到某一页面的概率.
   - 例如, 用户可能直接输入 IBM 的 URL, 而不是通过链接到达.

Information Extraction (IE)

核心概念

Information Extraction (IE) 的目标是从非结构化文本中提取细粒度的信息, 如特定类型的实体 (instances) 或实体之间的关系 (relationships).

- 类似于快速阅读一段文本, 找到关键的对象 (如名字、日期) 或它们之间的联系 (如谁与谁是同事).
- **应用领域**: 新闻摘要、产品信息提取、命名实体识别等.

Approaches to IE

1. **Finite State Automata (有限状态自动机)**:
   - 利用模式匹配技术提取特定属性.
   - **示例**: 从 "IBM ThinkBook 970. Our price: $399" 中提取:

```
Attributes: {
  Manufacturer = IBM,
  Model = ThinkBook 970,
  Price = $399.00
}
```

2. **Probabilistic Models（概率模型）**：
   - 使用统计方法捕捉文本中的不确定性。
   - 适合需要在大量文档中发现模式的任务。
3. **Conditional Random Fields（条件随机场）**：
   - 一种常用的序列标注方法，擅长提取连续文本中的实体和关系。
   - **示例**：在“John works at IBM”中标记：
     - John: PERSON
     - IBM: ORGANIZATION

Finite State Automata: Attribute-Based Extraction

原理

通过正则表达式或模板匹配，利用有限状态自动机从文本中提取特定的属性。

示例模板（正则表达式）：

- `[0-9]+`：匹配一个或多个数字。
- `[.] [0-9][0-9]`：匹配小数点后跟两位数字。
- `[$][0-9]+([.][0-9][0-9])?`：匹配货币值，如 `$249.99`。

🌐 应用

例如，从以下文本中提取价格信息：

- 文本：`The product costs $249.99 or $1.23 depending on size.`
- 提取结果：`249.99` 和 `1.23`。

📦 总结

Information Extraction 的本质是将非结构化的文本转化为结构化的信息，常见方法包括：

1. **规则匹配（Finite State Automata）**：快速且适合固定格式的任务。
2. **机器学习方法（CRF、Probabilistic Models）**：适合复杂或动态格式。

实体识别的正则表达式

示例 1：匹配电话号码

正则表达式：

$$([0-9]+[-\s])+[0-9]+$$

用于匹配电话号码，具体含义如下：

- `[0-9]+`：匹配一个或多个数字。
- `[-\s]`：匹配连字符 `-` 或空格。
- `([0-9]+[-\s])+`：匹配数字加上分隔符（连字符或空格）的模式，至少重复一次。
- `[0-9]+`：确保电话号码的末尾是数字。

可以匹配的示例：

- `0163-865-1125`
- `725 1234`
- `12-34-56`

无法匹配的示例：

- `+44 (0)163 865 1125`（因为包含了符号 `+` 和括号 `()`，正则表达式未定义这些符号）。

示例 2：匹配邮政编码（英国）

正则表达式：

$$[A-Z][A-Z]?[0-9][0-9]?\s?[0-9][A-Z][A-Z]$$

用于匹配英国风格的邮政编码，具体含义如下：

- `[A-Z]`：匹配一个大写字母。
- `[A-Z]?`：匹配零个或一个大写字母（可选）。
- `[0-9]`：匹配一个数字。
- `[0-9]?`：匹配零个或一个数字（可选）。
- `\s?`：匹配零个或一个空格（可选）。
- `[0-9][A-Z][A-Z]`：匹配一个数字，后面跟两个大写字母。

可以匹配的示例：

- `CW3 9SS`
- `SE5 0EG`
- `SE50EG`

无法匹配的示例：

- `M13 9PL`（格式不符合预期）。
- `M139PL`（中间缺少空格）。