候选词预测试验

基于 n-gram 语言模型的自动补全系统设计与问题：

预测逻辑：n-gram 模型通过统计历史 n-1 个词的条件概率预测下一个词。对于 4-gram（n=4），下一词w的概率计算为：

$$P(w|\text{"I want to eat"}) \approx \frac{Count(\text{"I want to eat"} + w)}{Count(\text{"I want to eat"})}$$

统计语料库中“I want to eat”后接各词的频次，以及这个词条本身出现的频次，按概率排序选取 top-k 候选词。

核心问题：

稀疏性：若词料库中本身词条出现的次数过少的话，可能导致分母接近于 0，导致 n-gram 模型失效；

长距离依赖：n-gram 仅仅依赖最近 n-1 个词，无法捕捉与前面较远语义的关联。

实现的 top-k 预测词：

首先附上执行代码后的结果：

```
Unigram 预测结果（最常见词）：[]
Bigram 预测结果（基于最后1个词 'eat' 前的 'to'）：['the', 'too', "'"]
Trigram 预测结果（基于最后2个词 'to eat' 前的 'want to'）：['his', 'up', '.']
```

原理阐述：

Unigram（1-gram）：只看当前词本身的频率，不考虑上下文，公式：

$$P(w) = \frac{Count(w)}{总词数}$$

Bigram(2-gram):看前一个词，预测下一个词，公式：

$$P(w_n|w_{n-1}) = \frac{Count(w_{n-1}, w_n)}{Count(w_{n-1})}$$

Trigram（3-gram）：看前两个词，预测下一个词，公式：

$$P(w_n|w_{n-2}, w_{n-1}) = \frac{Count(w_{n-2}, w_{n-1}, w_n)}{Count(w_{n-2}, w_{n-1})}$$

优缺点分析：

优点：简单易实现，适合小规模语料库快速构建；

缺点：稀疏性--语料库中没出现的 n-gram 会导致概率为 0，而且只能看最近 n-1 个

词，无法处理复杂语义；