作业一：实现N-gram语言模型平滑算法

1. 作业要求

基于提供的Python文件/Jupyter Notebook文件，以代码填空的形式，实现N-gram语言模型，并使用带Good-Turing折扣算法的Katz回退算法加以优化。填空完毕后，需在所给数据集上进行测试并展示结果。

提示：只需填写代码中 `TODO` 标记的空缺位置即可，代码量共计约30行；整个流程（训练和测试）的参考耗时约为30分钟，最终的平均PPL应小于200000。

2. 算法说明

2.1 带Good-Turing折扣算法的Katz回退算法

在课程中我们学习了N-gram语言模型和一种简单的（不带折扣的）回退算法的公式：

$$P(w_k|W_{k-n+1}^{k-1}) = \begin{cases} 1 * \dfrac{C(W_{k-n+1}^{k})}{C(W_{k-n+1}^{k-1})} & C(W_{k-n+1}^{k}) > 0 \\ 0.4 * P(w_k|W_{k-n+2}^{k-1}) & \text{否则} \end{cases}$$

所谓带Good-Turing折扣算法的Katz回退算法，即将上面常系数的折扣因子和回退因子改为可变的形式，如下所示公式：

$$P(w_k|W_{k-n+1}^{k-1}) = \begin{cases} d(W_{k-n+1}^{k}) * \dfrac{C(W_{k-n+1}^{k})}{C(W_{k-n+1}^{k-1})} & C(W_{k-n+1}^{k}) > 0 \\ \alpha(W_{k-n+1}^{k-1}) * P(w_k|W_{k-n+2}^{k-1}) & \text{否则} \end{cases}$$

该公式中 α 和 d 如何计算，本说明会在下面作简要推导、介绍。

2.2 计算回退因子 α

首先，不论选取何种 $\alpha(W_{k-n+1}^{k-1})$，我们始终应满足：

$$\sum_{w_k \in V} P(w_k|W_{k-n+1}^{k-1}) = 1$$

在此基础上，对已知的W_{k-n+1}^{k-1}，记V_+为对应N-gram在训练语料中出现过的词的集合：

$$V_+ = \{w_k | C(W_{k-n+1}^{k-1}, w_k) > 0\}$$

类似的，记V_-为对应N-gram在训练语料中未出现的词的集合：

$$V_- = \{w_k | C(W_{k-n+1}^{k-1}, w_k) = 0\} = V \setminus V_+$$

因此，我们有

$$\sum_{w_k \in V_+} P(w_k | W_{k-n+1}^{k-1}) + \sum_{w_k \in V_-} \alpha(W_{k-n+1}^{k-1}) P(w_k | W_{k-n+2}^{k-1}) = 1$$

可解得

$$\begin{aligned}\alpha(W_{k-n+1}^{k-1}) &= \frac{1 - \sum_{w_k \in V_+} P(w_k | W_{k-n+1}^{k-1})}{\sum_{w_k \in V_-} P(w_k | W_{k-n+2}^{k-1})} \\ &= \frac{1 - \sum_{w_k \in V_+} P(w_k | W_{k-n+1}^{k-1})}{1 - \sum_{w_k \in V_+} P(w_k | W_{k-n+2}^{k-1})}\end{aligned}$$

2.3 计算折扣因子 d

Good-Turing折扣（以下简称“折扣”）是一种将出现次数多的N-gram的概率摊给出现次数少的一种策略，具体地，是将训练数据中出现次数为 $r+1$ 次的N-gram的概率重新分配给出现次数为 r 次的N-gram。

考虑到该策略会导致出现次数最多的N-gram的概率变成零，带来很大的误差，因此实践中该策略只在低频N-gram上采用。可以选取频次 $\theta = 7$ 作为是否采用折扣策略的阈值。因此，对$w_k \in V_+$，可以写出如下公式：

$$d(W_{k-n+1}^{k-1}, w_k) = \begin{cases} 1 & C(W_{k-n+1}^{k-1}, w_k) > \theta \\ d'(W_{k-n+1}^{k-1}, w_k) & \text{否则} \end{cases}$$

此时，由于高频N-gram没有将概率匀出来给低频N-gram，因此直接应用原折扣策略，会导致加起来的概率和超过1。这可以应用一种插值策略来解决。

2.4 修正后的折扣因子 d'

注意到，折扣策略实质上就是将出现次数为 $(r+1)$ 次的N-gram的概率和 P_{r+1} ，摊给出现次数为 r 次的N-gram，因此可以将问题抽象一下：

折扣前，所有出现频率为 r 的N-gram的概率和为 P_r ，不插值直接折扣后变为 P_{r+1} 。

应用折扣策略前后，高频N-gram的概率和没有变化，因此零频N-gram与低频N-gram的概率和也不应因折扣而改变。注意到折扣策略会将P_1的概率和分给零频N-gram，因此折扣后，低频N-gram的概率和应为$\sum_{i=2}^{\theta} P_i$，即，插值策略应满足

$$\sum_{r=1}^{\theta}(\lambda P_{r+1} + (1-\lambda)P_r) = \sum_{r=2}^{\theta} P_i$$

易解得$\lambda = \frac{P_1}{P_1 - P_{\theta+1}}$。

由于

$$P_r = \sum_{C(W_{k-n+1}^{k-1}, w_k)=r} P(w_k|W_{k-n+1}^{k-1}) = N_r \frac{r}{C(W_{k-n+1}^{k-1})}$$

代入可得

$$\lambda = \frac{N_1}{N_1 - (\theta+1)N_{\theta+1}}$$

因此

$$d'(W_{k-n+1}^{k-1}, w_k) = \lambda \frac{(r+1)N_{r+1}}{rN_r} + (1-\lambda)$$

其中 $r = C(W_{k-n+1}^{k-1}, w_k)$，而 N_r 表示前缀为W_{k-n+1}^{k-1}且出现次数是 r 次的 N-gram 的种类数。

3. 提交方式

以下两种方式选择其一提交至CANVAS平台即可

- 直接提交带有已补全的代码及运行结果的Jupyter Notebook文件，文件中要求对所填代码作必要的说明。
- 提交已补全的Python文件和实验报告，报告中要求对代码作必要的说明并展示运行结果。

4. 文件说明

- `data` - 含有指定的训练与测试语料
- `ngram.ipynb` - 提供的Jupyter Notebook文件
- `ngram_h.py` - 提供的Python文件，内容与 `.ipynb` 中的代码相同

- `作业说明.pdf` - 本说明文档

5. 补充说明

提供的代码中采用了Python 3.5引入的类型注解语法，此处简要解释：

定义变量或声明函数参数时，可以采用

```
variable: Type
```

的方式标注变量类型。定义函数时，可以采用

```
def foo(a: A) -> B:
```

的方式声明参数与返回值类型。

类型注解不是强制的，在运行时不起作用。更详细的类型语法请自行查询Python文档与教程，此处不再介绍。