- 拼音输入法实验报告
  - 实验环境介绍
  - 使用的语料库和数据预处理方法
    - 使用的语料库组合及分析
    - 数据预处理方法
  - 基于字的二元模型的拼音输入法
    - 基本思路、公式推导、实现过程
    - 实验效果
    - 样例分析
      - 样例选取
      - 样例分析
    - 性能分析
    - 时间与空间复杂度分析
      - 时间复杂度
      - 空间复杂度
  - 实现的其他模型或算法
    - 在提出其他模型之前的思考
      - 小插曲
      - 针对性解决二元模型样例分析问题
      - Pilot Study
    - 三元模型
      - 基本思路、公式推导、实现过程
        - Ablation Study
      - 实验效果
      - 样例分析
      - 性能分析
      - 时间与空间复杂度分析
    - 新模型探索
      - 实现过程（Viterbi）：
      - 实验效果
      - 性能分析：
    - 模型间对比
  - 其他评价指标探究
  - 对实验的感受和建议


# 拼音输入法实验报告


姓名：万子豪

学号：2022012086


# 实验环境介绍

本次实验在wsl下进行，配置了Python>=3.9的虚拟环境

使用了的Python库：

- tqdm
- pypinyin

# 使用的语料库和数据预处理方法

使用的语料库组合及分析

使用的语料库组合：

1. 新浪新闻2016年的新闻语料库（以下简称**sina**）
2. 新浪新闻2016年的新闻语料库+微博情绪分类技术评测中通用训练集的微博语料库（简称**smp**）（二者组合简称**sina_and_smp**）

选用原因：

1. 分析语料库特点：
   - sina语料库较大，而smp比较小
   - sina语料库是新闻的采样，语言风格较为正式；smp语料库是微博发言的采样，语言风格比较随意，包含部分网络用语，与测试文件中的一些句子类似
2. 选取组合：类比大模型使用的预训练数据集和微调数据集，第二组将sina作为预训练数据集规范模型的基本语言，再在smp上进行微调来适应日常语言风格

使用了拼音汉字表，用于在推理时遍历拼音对应的汉字

未使用一二级汉字表，在权衡处理一二级汉字表速度和生僻字对于整体的统计次数的影响后选择了不使用一二级汉字表

数据预处理方法

对于**sina**中的每个**txt**文件：

1. 采用gbk编码打开
2. 对于每行，用json库解析，取出其中的html字段
3. （可选，记为**parse**，在二元模型的实验中默认不选）将段落按照标点符号分成多个字段
4. 采用正则表达式匹配其中的所有中文，合成一句话，放入句子列表

对于**smp**中的每个**txt**文件：

1. 采用gbk编码打开，观察发现没有主段落外的多余中文干扰（如sina中有title），故不再用json解析
2. （可选，记为**parse**，在二元模型的实验中默认不选）将段落按照标点符号分成多个字段
3. 采用正则表达式匹配其中的所有中文，合成一句话，放入句子列表

处理句子列表： 对于其中的每个句子，遍历统计出现的一元字、二元字、三元字（可选，记为**three**，默认不选）、间隔二元字（可选，记为**skip2**，默认不选），将(某类型字, 出现次数)键值对记录到某类型对应的字典中

处理拼音汉字表： 对于拼音汉字表中的每一行，将(拼音, 对应汉字list)键值对记录到拼音汉字字典中

处理多音字（可选，记为**emit**，默认不选）：

1. 采用了pypinyin库，为每个句子列表中的句子注音
2. 将（字，{读音：频率}）键值对记录到发射概率字典中

注：对于一些较大的语料库，若不选parse，可能导致中间文件过大而无法继续推理

# 基于字的二元模型的拼音输入法

## 基本思路、公式推导、实现过程

基本思路及公式推导（参考课程ppt）：

1. 对于拼音序列O到句子S的问题，通过条件概率公式 $P(S|O) = P(S)P(O|S)/P(O)$ 转化，忽略常数部分，得到优化目标：使 $P(S) = \prod P(w_i|w_1...w_{i-1})$ 最大
2. 只考虑二元语法，使 $P(S) = \prod P(w_i|w_{i-1})$ 最大
3. 相当于使 $-\Sigma log(P(w_i|w_{i-1}))$ 最大
4. 将其 $-log(P(w_i|w_{i-1}))$ 转化为Viterbi算法中的转移概率，从而化为一个最短路径问题

实现过程（Viterbi）：

1. 对于每一个拼音序列，从前到后进行遍历每个拼音
2. 对第i个拼音 $x_i$ ，分类讨论：
    1. 若i==0，通过拼音汉字字典找到拼音 $x_i$ 对应的所有汉字 $word_i$ ，通过一元字表获取该字出现次数 $wordcount_i$ ，计算所有一元语法概率 $wordcount_i/totalwordcount$ ，作为 $dp[0][word_0]$
    2. 若i>0:
        1. 通过拼音汉字字典找到拼音 $x_i$ 对应的所有汉字 $word_i$ ，通过一元字表获取该字出现次数 $wordcount_i$
        2. 枚举每个上一个字 $x_{i-1}$ ，通过二元字表获取二元字 $x_{i-1}x_i$ 出现次数 $twowordcount_i$ ，通过一元子表获取一元字 $x_{i-1}$ 出现次数 $lastwordcount_i$
        3. 计算所有二元语法概率 $score = -log(\lambda_2 * twowordcount_i/lastwordcount_i + (1-\lambda_2)wordcount_i/totalwordcount)$ ，采用了一元语法做平滑，其中 $\lambda_2$ 为超参数
        4. 使用 $max(dp[i-1][x_{i-1}] + score)$ 更新 $dp[i][x_i]$ ，记录成功更新 $x_i$ 的前驱 $x_{i-1}$
        5. 对于二元语法概率公式中有任何 $count = 0$ 的情况，说明其出现频率很低，设置 $score$ = BIGNUM = 1000000，保证不会影响dp过程
3. 在遍历完所有拼音后，取 $word_{-1}$ 使 $dp[-1][word_{-1}]$ 最大，然后反复取其前驱加入字符串 $reverseans$ ，直到取到句首，再翻转 $reverseans$ 得到 $finalans$

注：为何要将条件概率中的 $P(twoword)/P(lastword)$ 转为 $twowordcount_i/lastwordcount_i$ ：

- 计算 $P(twoword)$ 需要额外统计 $totaltwowordscount$ ，而花费额外时间
- $P(twoword)/P(lastword) = twowordcount_i/lastwordcount_i * totalwordcount/totaltwowordscount$ ，而 $totalwordcount/totaltwowordscount$ 约为 **总字数/（总字数－总句子数）** ，接近于1，可以忽略

# 实验效果

给定测试样例上的准确率：

- 句准确率（sentence acc）：0.387
- 词准确率（word acc）：0.853

训练时间（processing time）：320s

生成一句话的平均时间：0.0113s

生成所有给定测试样例的总时间（inference time）：5.68s

# 样例分析

样例选取

对于n元语法，在n较小时容易丢失句首部分的信息，导致输出句子异常，故长句子应该是输入法提升正确率的难点。故以下挑选了正确和错误的各五个长句，错误长句对中用*粗斜体*标出了错误部分：

正确长句：

经济建设和文化建设突出了十八大精神的重要性

社会主义阶段的最根本的任务就是发展生产力

要把我们的祖国建设成为一个富强民主的国家

才发现自己没有想好要做什么成为怎样的人

大量餐厨垃圾与其他垃圾混合填埋或焚烧

错误长句：

他感到太幸福了但他一点也不骄傲因为一颗好的心是永远不会骄傲的

他*赶*到太幸福了但他一点也不骄傲因为一颗好的心是永远不会骄傲的

中国共产党员的初心和使命是为中国人民谋幸福为中华民族谋复兴

中国共产党员的初心和使命是为中国人民谋幸福为中华民族谋*副省*

这世上只有一种成功就是能够用自己喜欢的方式度过自己的一生

这*是*上只有一种成功就是能够用自己喜欢的方式都过自己的*医*生

我们中有些人注定要在日常生活的点滴中寻找生命的意义

我们中有些人注定要在日常生活的*典的*中寻找生命的意义

我国在外交中一向以和平共处五项原则为基本准则

我国在外交中一*项议*和平共处五项原则为基本准则

样例分析

对于正确长句，我们大致可以看出其二元字内部的关联度比较高

而我们可以从错误长句的样例中看出当前输入法的诸多不足：

- Autoregressive的方式造成其无法看到下文信息，很多字仅凭上文无法成功推理，例：“他*赶*到太幸福了”，“这*是*上只有一种成功”
- 对于n=2的n-gram，丢失上文信息过多，例：“过自己的*医*生”
- 推理受到多音字干扰，例：“*副省*”，“*典的*”

在**实现的其他模型或算法**部分将针对这些不足进行改进

# 性能分析

对于基础二元模型，主要可以调整的超参数有计算二元概率时的 $score = -log(\lambda_2 * twowordcount_i/lastwordcount_i + (1-\lambda_2)wordcount_i/totalwordcount)$ 中的 $\lambda_2$

这里固定语料库为sina，对于不同的 $\lambda 2$，训练时间均约为320s，推理时间均约为6s，选取了表现较好的一些 $\lambda_2$

| $\lambda_2$ | **0.9** | **0.99** | **0.999** | **0.9999** | **0.99999** |
|---|---|---|---|---|---|
| sentence acc | 0.381 | 0.385 | 0.387 | 0.387 | 0.387 |
| word acc | 0.853 | 0.853 | 0.853 | 0.853 | 0.853 |

可以看到总体趋势为性能随着平滑项 $wordcount_i/totalwordcount$ 的比例下降逐渐提升，并在0.999附近大致收敛

二元语法相对于一元语法提升的句子连贯性在这里的定量实验中有所体现

# 时间与空间复杂度分析

时间复杂度

对于长度为n的每个拼音列表，需要遍历其中的每个拼音，列举拼音对应的汉字，并枚举上一个拼音对应的汉字

假设拼音平均对应汉字数为m，则时间复杂度为 $O(n*m^2)$

计算拼音汉字表中拼音平均对应汉字数为18.5，计算标准输入中拼音列表的平均长度为10.4，假设1秒进行$10^9$次运算，则处理每个句子的平均时间为 $10.4 * (18.5^2) / 10^9 = 0.00000356s$

对比发现其远小于实际测试的生成一句话的平均时间 $0.0113s$，通过打印不同代码块运行的时间，我发现：

- 加载多个数据字典占据了程序运行的很长时间（约2s）
- 文件输入输出也占据了一定时间
- 对于每个二元字的计算不能简单看作单次运算

以上两点造成了不精准的时间测量和时间估计

空间复杂度

空间复杂度主要来源于加载的多个字典，包括一元字字典（假设键值对个数为 $N_1$）、二元字字典（假设键值对个数为 $N_2$）、拼音汉字表（假设拼音个数为p，拼音平均对应汉字数为m）；以及Viterbi算法用到的dp二维数组（假设拼音列表平均长度为n、拼音平均汉字数为m）

其中二元字组合会显著多余一元字，故 $N_2 >> N_1$ ，总空间复杂度 $O(N_2 + p * m + n * m)$

# 实现的其他模型或算法

## 在提出其他模型之前的思考

小插曲

在实现二元模型的过程中，我曾经不小心把计算二元转移概率的公式 $score = -log(\lambda_2 * twowordcount_i/lastwordcount_i + (1-\lambda_2)wordcount_i/totalwordcount)$ 写成了 $score = -log(\lambda_2 * twowordcount_i/currentwordcount_i + (1-\lambda_2)wordcount_i/totalwordcount)$ ，却在OJ上仍然拿到了31的分数，并很久都没有发现问题

发现BUG后，我开始思考为什么写成 $twowordcount_i/currentwordcount_i$ 仍能拿到不少分数，这个变种的概率是否也有一定的意义？

正确的概率代表二元字 $x_{i-1}x_i$ 在前一个字 $x_{i-1}$ 出现时的概率，而错误的概率代表二元字 $x_{i-1}x_i$ 在后一个字 $x_i$ 出现时的概率，也帮助建立了二元字内部的连贯性

进一步的，到了三元模型，“错误”的概率是否也有意义？

我观察到三元模型中三元字 $x_{i-2}x_{i-1}x_i$ 在前后两个字组成的间隔二元字 $x_{i-2} < space > x_i$ 出现时的概率似乎与上方提到的部分又不相同，似乎与“已知前后文向中间填空”这样一种任务有同样的形式（虽然被简化了很多）

而我就联想到了著名预训练模型**BERT**的训练目标：混合**auto-regressive**和**mask-filling**

因此，我在后续的新模型探索中使用了这种填空概率（后文中记为 $threewordcount_i/skiptwowordcount_i$ ），发现真的有所提高；某种程度上，这也对应了样例分析中无法看到下文信息的问题

有关填空概率的实验，详见新模型探索部分

针对性解决二元模型样例分析问题

在二元模型的样例分析中，曾找到三个表现不佳的原因：

- 无法看到下文信息
- 上文长度过短
- 多音字无法辨别

无法看到下文信息的解决方案在上面的填空概率的思考中

对于上文长度过短的缺陷，可以引入三元模型来解决：引入条件概率 $threewordcount_i/lasttwowordcount_i$ 将转移概率延伸至三元范围

对于多音字无法辨别的缺陷，我在查询Viterbi算法时看到了变种，引入发射概率（**emit**），即加入当前这个字是当前这个读音的几率的权重（参考了 https://zh.wikipedia.org/wiki/%E7%BB%B4%E7%89%B9%E6%AF%94%E7%AE%97%E6%B3%95）

在数据预处理部分：

对于语言风格，我发现测试语料中的风格更接近SMP而不是新闻数据库，故引入**SMP**也会加强表现

对于是否把段落分割成句子，根据测试语料的平均长度可以推测分割（**parse**）更好

**Pilot Study**

| Strategy | w/o emit; w/o smp; w/o parse (basic version) | w emit; w/o smp; w/o parse | w/o emit; w smp; w/o parse | w/o emit; w smp; w/o parse |
|---|---|---|---|---|
| sentence acc | 0.387 | 0.417 | 0.389 | 0.395 |
| word acc | 0.853 | 0.868 | 0.853 | 0.853 |

可以看到，这几种强化方案均在提升句准确率上达到了一定效果

# 三元模型

基本思路、公式推导、实现过程

三元模型的思路与公式推导与二元模型类似

基本思路及公式推导：

1. 对于拼音序列O到句子S的问题，通过条件概率公式 $P(S|O) = P(S)P(O|S)/P(O)$ 转化，忽略常数部分，得到优化目标：使 $P(S) = \prod P(wi|w1...wi-1)$ 最大
2. 考虑三元语法，使 $P(S) = \prod P(w_i|w_{i-1}w_{i-2})$ 最大
3. 相当于使 $-\Sigma log(P(w_i|w_{i-1}w_{i-2}))$ 最大
4. 将 $-log(P(w_i|w_{i-1}w_{i-2}))$ 转化为Viterbi算法中的转移概率，从而化为一个最短路径问题

实现过程（Viterbi）：

1. 对于每一个拼音序列，从前到后进行遍历每个拼音
2. 对第i个拼音 $x_i$ ，分类讨论：
    1. 若i==0，通过拼音汉字字典找到拼音 $x_i$ 对应的所有汉字 $word_i$ ，通过一元字表获取该字出现次数 $wordcount_i$ ，计算所有一元语法概率 $wordcount_i/totalwordcount$ ，作为 $dp[0][word_0]$
    2. 若i==1:
        1. 通过拼音汉字字典找到拼音 $x_i$ 对应的所有汉字 $word_i$ ，通过一元字表获取该字出现次数 $wordcount_i$
        2. 枚举每个上一个字 $x_{i-1}$ ，通过二元字表获取二元字 $x_{i-1}x_i$ 出现次数 $twowordcount_i$ ，通过一元子表获取一元字 $x_{i-1}$ 出现次数 $lastwordcount_i$
        3. 计算所有二元语法概率 $score = -log(\lambda_2 * twowordcount_i/lastwordcount_i + (1-\lambda_2)wordcount_i/totalwordcount)$ ，其中 $\lambda_2$ 为超参数
        4. 将score乘上发射概率：$score = score * duoyinscore$
        5. 使用 $max(dp[i-1][x_{i-1}] + score)$ 更新 $dp[i][x_i]$ ，记录成功更新 $x_i$ 的前驱 $x_{i-1}$
        6. 对于二元语法概率公式中有任何 $count = 0$ 的情况，说明其出现频率很低，设置 $score$ = BIGNUM，保证不会影响dp过程
    3. 若i>=2:
        1. 通过拼音汉字字典找到拼音 $x_i$ 对应的所有汉字 $word_i$ ，通过一元字表获取该字出现次数 $wordcount_i$

2. 枚举每个上一个字 $x_{i-1}$，再取出上一个字的前驱 $x_{i-2}$ ，通过三元字表获取三元字 $x_{i-2}x_{i-1}x_i$ 出现次数 $threewordcount_i$，通过二元字表获取二元字 $x_{i-1}x_i$ 出现次数 $currenttwowordcount_i$ 和 $x_{i-2}x_{i-1}$ 出现次数 $lasttwowordcount_i$，通过一元子表获取一元字 $x_{i-1}$ 出现次数 $lastwordcount_i$
3. 计算所有三元语法概率 $score = -log(\lambda_3 * threewordcount_i/lasttwowordcount_i + (1-\lambda_3) * (\lambda_2 * twowordcount_i/lastwordcount_i + (1-\lambda_2)wordcount_i/totalwordcount))$ ，采用了二元和一元语法做平滑，其中 $\lambda_2$、$\lambda_3$ 为超参数
4. 使用 $max(dp[i-1][x_{i-1}] + score)$ 更新 $dp[i][x_i]$ ，记录成功更新 $x_i$ 的前驱 $x_{i-1}$
5. 对于 $threewordcount_i$ 或 $lasttwowordcount_i$ 等于零的情况，只计算二元语法（相当于设置 $\lambda_3 = 0$）
6. 对于二元语法概率公式中有任何 $count = 0$ 的情况，设置 $score$ = BIGNUM，保证不会影响dp过程

3. 在遍历完所有拼音后，取 $word_{-1}$ 使 $dp[-1][word_{-1}]$ 最大，然后反复取其前驱加入字符串 $reverseans$ ，直到取到句首，再翻转 $reverseans$ 得到 $finalans$

以上三元语法模型的实现中有两点比较不intuitive，以下是解释：

- 没有枚举所有的 $x_{i-2}$，称为**enum**：若枚举，时间复杂度会提升至 $O(n * m^3)$，造成显著的推理时间增加
- 无法直接计算三元转移概率时尝试退化为二元，称为**degrade**：
  - 由于语料库不够大，语料库中的三元字对于测试语料中的三元字覆盖率可以预见的低，无法计算三元转移概率会是常态，若不进行退化而直接设置BIGNUM容易给出不经实际频率推理的结果
  - 退化为二元是否不公平：假设 $\lambda_3 = 0.9, \lambda_2 = 0.9$ ，退化成二元的 $y_{i-2}y_{i-1}y_i$ 的得分若大于三元 $x_{i-2}x_{i-1}x_i$，即为 $(0.9 * twowordcountY_i/lastwordcountY_i + 0.1 * wordcountY_i/totalwordcountY) > (0.9 * threewordcountX_i/lasttwowordcountX_i + 0.1 * (0.9 * twowordcountX_i/lastwordcountX_i + 0.1 * wordcountX_i/totalwordcountX))$，忽略较小的项，再移项得 $twowordcountY_i/lastwordcountY_i > *threewordcountX_i/lasttwowordcountX_i$，此式的成立要求两个二元语法的出现次数乘积比一元语法出现次数还高，这不太容易达成，除非Y中的两个字真的有极强关联，故这种处理是相对公平的

以下的实验发现以上两点确实对于三元语法模型的速度和准确率都有提升

#### Ablation Study

实验中 $\lambda_3 = 0.8, \lambda_2 = 0.97$，采用了发射概率（emit），数据处理时进行了段落分割（parse），使用sina_and_smp数据库

| Strategy | w/o enum; w degrade; (current version) | w enum; w degrade | w/o enum; w/o degrade |
|---|---|---|---|
| sentence acc | 0.621 | 0.485 | 0.592 |
| word acc | 0.916 | 0.894 | 0.867 |
| time | 31.98 | 198.0 | 33.50 |

意外发现不枚举对于准确率也有提升，推测是因为字与其前驱间包含了丰富的全句信息

#### 实验效果

采用了最佳超参数 $\lambda_3 = 0.74, \lambda_2 = 0.98$，将在性能分析部分展示

给定测试样例上的准确率：

- 句准确率（sentence acc）：0.625
- 词准确率（word acc）：0.916

训练时间（processing time）：2400s（多音字的统计用到了pypinyin库，显著提高了训练时间）

生成一句话的平均时间：0.0730s

生成所有给定测试样例的总时间（inference time）：36.59s

样例分析

同样挑选了正确和错误的各五个长句，错误长句对中用*粗斜体*标出了错误部分：

正确长句：

他感到太幸福了但他一点也不骄傲因为一颗好的心是永远不会骄傲的

中国共产党员的初心和使命是为中国人民谋幸福为中华民族谋复兴

我们中有些人注定要在日常生活的点滴中寻找生命的意义

本次普查活动有助于帮助同学们走出心理误区

毛泽东思想和中国特色社会主义理论体系概论

错误长句：

这世上只有一种成功就是能够用自己喜欢的方式度过自己的一生

这*是*上只有一种成功就是能够用自己喜欢的方式度过自己的*医*生

我国在外交中一向以和平共处五项原则为基本准则

我国在外交中一*项*议和平共处五项原则为基本准则

郑和下西洋为青花瓷的迅速崛起提供了历史契机

*整合*下西洋为青花瓷的迅速崛起提供了历史契机

经济建设和文化建设突出了十八大精神的重要性

经济建设和文化建设突出了十八大精*深*的重要性

地道战是一种以地道为策略应用的陆军步兵战术

地道战是一种以*低到位测*略应用的陆军步兵战术

分析：

部分正确长句来自二元模型中的错误长句，我们可以看到其中的多音字、长上文问题有所改善

而错误长句也展现出不同程度的上下文问题

性能分析

对于三元模型，主要可以调整的超参数有计算三元概率时的 $score = -log(\lambda_3 * threewordcount_i/lasttwowordcount_i + (1-\lambda_3) * (\lambda_2 * twowordcount_i/lastwordcount_i + (1-\lambda_2)wordcount_i/totalwordcount))$ 中的 $\lambda_2$ 和 $\lambda_3$

这里固定语料库为sina_and_smp，采用发射概率，进行段落分割，对于不同的 $\lambda_2$ 和 $\lambda_3$，训练时间均约为2400s，推理时间均约为35s，选取了表现较好的一些 $\lambda_2$ 和 $\lambda_3$

| $\lambda_3$ and $\lambda_2$ | **0.74, 0.98** | **0.74, 0.96** | **0.74, 0.95** | **0.74, 0.99** | **0.74, 0.97** | **0.75, 0.97** | **0.73, 0.97** | **0.65, 0.97** | **0.7, 0.97** | **0.75, 0.97** | **0.85, 0.97** | **0.9, 0.97** | **0.8, 0.97** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| sentence acc | 0.625 | 0.625 | 0.621 | 0.619 | 0.625 | 0.625 | 0.625 | 0.621 | 0.623 | 0.623 | 0.615 | 0.589 | 0.621 |
| word acc | 0.916 | 0.915 | 0.916 | 0.917 | 0.916 | 0.916 | 0.915 | 0.915 | 0.915 | 0.916 | 0.916 | 0.911 | 0.916 |

可以看到三元概率的比例逐渐提升不会随模型表现一直变好，在0.74左右达到最高峰，这说明二元的连贯性在句子的整体含义中占有重要地位；而一元语法因为缺乏对前文的注意难以作为重点

**时间与空间复杂度分析**

时间复杂度：

由于上述提到的三元模型没有枚举 $x_{i-2}$ 而直接选择前驱，所以时间复杂度与二元模型相同，为 $O(n*m^2)$

空间复杂度：

空间复杂度主要来源于加载的多个字典，包括一元字字典（假设键值对个数为 $N_1$）、二元字字典（假设键值对个数为 $N_2$）、三元字字典（假设键值对个数为 $N_3$）、拼音汉字表（假设拼音个数为p，拼音平均对应汉字数为m）、多音字字典（假设键值对个数为 $N_4$）；以及Viterbi算法用到的dp二维数组（假设拼音列表平均长度为n、拼音平均汉字数为m）

其中$N_3$ 显著大于其他字典，总空间复杂度 $O(N_3+p*m+n*m)$

# 新模型探索

在提出其他模型之前的思考部分中，提到了填空概率的想法，这一部分即对填空概率进行实验

**实现过程（Viterbi）：**

仅把三元模型的实现中的 $\lambda_3*threewordcount_i/lasttwowordcount_i$ 替换为 $\lambda_3*(\lambda_{33}*threewordcount_i/lasttwowordcount_i+\lambda_{32}*threewordcount_i/skiptwowordcount_i+\lambda_{31}*threewordcount_i/currenttwowordcount_i)$，其中 currenttwoword 为 $x_{i-2}x_{i-1}x_i$ 的后两个字（虽然不像填空概率那么对应实际目标，但或许有意义），$\lambda_{31}+\lambda_{32}+\lambda_{33}=1$

**实验效果**

基于三元最佳超参数 $\lambda_3=0.74,\lambda_2=0.98$，同时选择了$\lambda_{31}=0.0005$、$\lambda_{32}=0.0005$、$\lambda_{33}=0.999$，数据见性能分析

给定测试样例上的准确率：

- 句准确率（sentence acc）：0.629
- 词准确率（word acc）：0.918

训练时间（processing time）：2700s（多音字的统计用到了pypinyin库，显著提高了训练时间，额外统计的间隔二元字也提高了训练时间）

生成一句话的平均时间：0.0766s

生成所有给定测试样例的总时间（inference time）：38.40s

前面的三元模型相当于 $\lambda_{33}=1,\lambda_{32}=\lambda_{33}=0$，这个新模型只对参数进行了改动，故以下只进行性能分析

**性能分析：**

这里固定语料库为sina_and_smp，采用发射概率，进行段落分割，$\lambda_2=0.98$ $\lambda_3=0.74$，对于不同的 $\lambda_{31}$、$\lambda_{32}$、$\lambda_{33}$，训练时间均约为2700s，推理时间均约为40s，选取了表现较好的一些 $\lambda_{31}$、$\lambda_{32}$、$\lambda_{33}$

| $\lambda_{31}$：$\lambda_{32}$：$\lambda_{33}$ | **0.5:0.5:999** | **0.6:0.4:999** | **0.4:0.6:999** | **0.7:0.3:999** | **0.3:0.7:999** | **0.9:0.1:999** | **0.1:0.9:999** | **0.5:0.5:9999** |
|---|---|---|---|---|---|---|---|---|
| sentence acc | 0.629 | 0.627 | 0.627 | 0.625 | 0.627 | 0.621 | 0.625 | 0.625 |
| word acc | 0.918 | 0.918 | 0.918 | 0.917 | 0.918 | 0.916 | 0.917 | 0.916 |

可以看出，填空概率配比等于或高于采用后两个字作为条件的无意义概率时，模型表现较好，证明了这个填空目标的合理性

# 模型间对比

这部分将以上每个部分中最有代表性的模型进行对比

| **Model** | 二元模型**(basic)** | 三元模型**(strongest)** | 采用填空概率增强的三元模型**(SOTA)** |
|---|---|---|---|
| sentence acc | 0.387 | 0.625 | 0.629 |
| word acc | 0.853 | 0.916 | 0.918 |
| inference time | 5.68s | 36.59s | 38.40s |
| processing time | 320s | 2400s | 2700s |

其中模型的具体参数：

- 二元模型：语料库为sina
- 三元模型：语料库为sina_and_smp，采用发射概率，进行段落分割，
- 采用填空概率增强的三元模型：语料库为sina_and_smp，采用发射概率，进行段落分割，有额外的填空概率

可以看到，在时间充裕的情况下，采用这里的后两种模型可以收获相对可观的准确率；而在时间有限制时，推荐采用去掉emit的三元模型，达到时间与准确率的平衡

附：不同参数数据处理时间表（由于加入emit后其时间变为主要部分，同列时间相近，故没有进行完全统计）

| | **default** | **three** | **emit** | **three+skip2** | **emit+three** | **emit+three+skip2** |
|---|---|---|---|---|---|---|
| **sina** | 316.37 | 777.85 | 2037.63 | 1018.59 | 2528.51 | 2548.38 |
| **sina_and_smp** | 332.51 | 752.10 | 1972.62 | 884.18 | 2390.13 | 2674.19 |
| **sina+parse** | 240.17 | 512.87 | - | 744.84 | - | - |
| **sina_and_smp+parse** | 232.87 | 510.99 | - | 744.88 | - | - |

# 其他评价指标探究

在实验中，主要指标为句准确率、词准确率（由于多数模型都很高故不重要）和生成句子所用的时间；对于比较简单的扩展如准确率与时间相除，因为两个量的标准不同故其绝对数值没有太多实际意义，选取模型时根据时间限制选取较高准确率的即可

而有关更多指标，我通过任务本身的性质出发，联想到从拼音列表到汉字句子的过程好像与翻译过程有些类似

于是，我尝试了机器翻译中的常用指标，BLEU和ROUGE-L

在实验中，我选了一些例子进行实验后，发现BLEU和ROUGE-L均与句准确率完全相等

通过研究BLEU、ROUGE的公式和一些计算例子后，我发现BLEU和ROUGE主要对机器翻译场景中乱序但正确的短语进行了处理，但输入法这个问题中由于每个拼音都有对应的汉字选项，基本不会出现乱序的问题，故这两个评价指标并不适用

最终，本实验没有采用除句准确率、词准确率和生成句子所用时间外的更多评价指标

# 对实验的感受和建议

在这个实验中，一些错误的尝试引发了我对于n-元语法、Autoregressive、Viterbi算法等诸多概念的更加深入的了解。通过设计Pilot Study、Ablation Study还有一系列参数调优过程，我收获了研究多变量未知问题的重要经验，并对课程的第一章产生了新的理解。虽然本次实验中还会有问题没有覆盖到，但提出新想法并成功验证的过程让人感到了强烈的成就感。