计算语言学作业 4 作业报告

解小锐 2019211105

2020 年 11 月 25 日

1 语料划分以及所分配 smoothing 策略说明

1.1 语料划分以及版本选取

我所分配到的语料编号为 s_1，选取了 **corpus-new** 版本的语料

1.2 所分配 smoothing 策略

我所分配到的三个 smoothing 策略分别是:

- laplace's law
- cross-validation
- simple-good-turing

2 工程文件说明

- data 文件夹: 存放一些计算的中间结果，减少计算的重复，提高 debug 效率
- config.py: 定义一些公共变量和函数
- text_tackle.py: 定义对文本的处理方法
- mylaplace.py:laplace'law 的实现
- mydeleted.py:cross-validation 的实现

- mysgt.py:simple-good-turing 的实现
- question1.py: 对问题 1 的实现
- question2_3.py: 对问题 2 和 3 的实现

3 三种方法的具体实现细节

- laplace's law: bigram 的概率计算使用 bigram 的平滑公式，而对应的条件概率则直接使用 laplace 的针对条件概率的平滑公式
- cross-validation: bigram 的概率计算和 statistical NLP 中的细节略有不同 (感觉 statistical NLP 也没把具体公式讲清楚，而且这种方法的 1985 年的原文也下载不下来，所以自己稍微 DIY 了方法)，对于一个 bigram，它在 retained 和 heldout 计算出来的 r* 进行加权平均，权重是 retained 和 heldout 的 N0 和 N1 占整个 N 的比例，然后条件概率是通过 bigram 求和得到 unigram 概率然后计算条件概率。注意在求出所有出现的 bigram 概率后，用 1-sum(P(seen bigram)) 求出剩下没出现的 bigram 的概率和，然后除以它们的数量。这是受到 simple-good-turing 的启发，而且 sgt 和 cross-validation 确实有很大联系：它们都假设频度相同的词的概率也相同！
- simple-good-turing: 具体参考原论文 Good-Turing Smoothing Without Tears，以及 github 上多个 simple-good-turing 的实现，我自己找到了三个，其中一个还是个 python 包，在求出 bigram 后，同样的把 bigram 求和求 unigram 的概率并继而求出条件概率

4 问题 1：计算 spearman 等级相关系数以及简单分析

4.1 spearman 相关系数计算结果

- laplace's law 与 cross-validation: **0.791731**
- laplace's law 与 simple-good-turing: **1.0**
- cross-validation 与 simple-good-turing: **0.791731**

4.2 例子分析

由于 laplace's law 方法和 simple-good-turing 方法的 spearman 系数为 1，所以这里主要分析其中一种方法与 cross-validation 方法的结果差异。对于 cross-validation 方法，1给出了它与 simple-good-turing 方法排序变化最大的前 3 个 bigram。因为剩下的排序变化大的 bigram 的相关特征与这 3 个类似，所以这里只列举 3 个。对于这 3 个 bigram，可以看到它们在使用 cross-validation 方法修正后的 r^* 与 simple-good-turing 方法的结果相比更小，距离实际的频次 2 相比更小，进一步分析原因后发现这 3 个 bigram 都有个共同特点：在 train set 和 valid set 中出现的频率都为 1，进一步计算相应的 $\frac{T_1^{01}}{N_1^0}$ 以及 $\frac{T_1^{10}}{N_1^1}$ 后发现，$\frac{T_1^{01}}{N_1^0} = \frac{59764}{219167} = 0.27$，而 $\frac{T_1^{10}}{N_1^1} = \frac{53674}{178720} = 0.30$，说明在 train set 中出现频度为 1 的 bigram 在 valid set 中出现的次数和相对于 train set 中频度为 1 的 bigram 出现的次数和太少，反过来也是一样，我认为这是 cross-validation 的缺陷之一，二分的 cross-validation 方法使用两次验证所得的概率 (频度）的加权平均来估计整体的概率 (频度)，然而如果划分后的集合的概率分布与原始集合的概率分布差异较大时，这种方法产生的误差就比较大。

表 1: cross-validation 方法与 simple-good-turing 排序变化最大的 3 个 bigram

bigram	d^2	unnormed r	cross-validation's r^*	simple-good-turing's r^*
(' 增加', ' 资金')	29344889112.25	2	0.5728	0.8600
(' 国家', ' 控股')	29344889112.25	2	0.5728	0.8600
(', ', ' 搞活')	29344889112.25	2	0.5728	0.8600

5 问题 2：构造句子并计算概率值和困惑度以及特定位置上的词的预测问题

5.1 句子 1

年轻人 不 讲 **武德**

不同 smoothing 方法预测的句子概率如2所示，对于**武德**位置上的预测概率

最大的前 10 个词如3所示，出现**武德**的概率如4所示。

表 2: 第一个句子的概率的概率和困惑度

方法	概率	困惑度
laplace's law	4.99277313856443e-23	5211.263435578822
cross-validation	4.725843804398451e-21	2441.1067564482973
simple-good-turing	7.327205124233939e-21	2269.0476093175353

表 3: " 武德" 位置上的预测概率最大的前 10 个词及其概率

词	laplace 概率	词	cross-validation 概率	词	simple-good-turing 概率
，	0.0006923981128125855	，	0.14754253540004467	，	0.18341310830108137
文明	0.0004991707324927942	礼貌	0.14349084250060848	文明	0.12939889190916978
礼貌	0.0003381479155596348	”	0.11614557335908854	礼貌	0.08444165455629515
的	0.00032204563386631883	文明	0.11589178329391145	的	0.07995178310631665
”	0.000273738788786371	的	0.061343868511482325	”	0.06649377444841421
政治	0.00017712509862647538	政治	0.028221913501479668	政治	0.03967694730391434
了	0.00016102281693315942	了	0.024098203762257814	了	0.035233036941883955
得	0.0001127159718532116	过	0.014414945565044444	得	0.022006618276631747
过	0.0001127159718532116	社会主义	0.014314345017767825	过	0.022006618276631747
学习	9.661369015989566e-05	是	0.014314345017767825	学习	0.017661548977294647

5.2 句子 2

扶贫 开发 工作 取得 很 **大** 成绩 。

不同 smoothing 方法预测的句子概率如5所示，对于**大**位置上的预测概率最大的前 10 个词如6所示，出现**大**的概率如7所示。

表 4: " 武德" 的概率和位置

方法	概率	位置
laplace's law	1.6102281693315944e-05	37271
cross-validation	3.428628387307575e-07	37271
simple-good-turing	3.8429031387605635e-07	37271

表 5: 第二个句子的概率的概率和困惑度

方法	概率	困惑度
laplace's law	2.5258462571347706e-36	3628.7688827868424
cross-validation	1.0246124026088542e-27	499.9701073604845
simple-good-turing	7.686190516942297e-28	514.5515437371423

表 6: " 大" 位置上的预测概率最大的前 10 个词及其概率

词	laplace 概率	词	cross-validation 概率	词	simple-good-turing 概率
大	0.002136002805496222	大	0.1631855550305357	大	0.18093680271072074
好	0.0011158223610801161	好	0.09622775172745045	好	0.09300846210475804
难	0.0009404788471960978	难	0.07750833668472314	难	0.07789896051562765
有	0.000701374055536073	有	0.05489671025394031	有	0.057299667728517856
不	0.0005738514999840597	不	0.046224959544430004	不	0.046317566967593066
少	0.0004782095833200497	少	0.036329655093739636	少	0.038084745074437244
强	0.0003506870277680365	快	0.0243788526355394	强	0.0.027117632745239453
快	0.0003347467083240348	强	0.024082391500419468	快	0.025748115812179617
高	0.00028692574999202985	高	0.01948500520073608	高	0.02164260620454463
重要	0.0002550451111040265	重要	0.017343611309033103	重要	0.018909077486134653

表 7: " 大" 的概率和位置

方法	概率	位置
laplace's law	0.002136002805496222	1
cross-validation	0.1631855550305357	1
simple-good-turing	0.18093680271072074	1

5.3 句子 3

我 **相信** 我 就是 我

不同 smoothing 方法预测的句子概率如8所示，对于**相信**位置上的预测概率最大的前 10 个词如9所示，出现**相信**的概率如10所示。

表 8: 第三个句子的概率的概率和困惑度

方法	概率	困惑度
laplace's law	3.349342962540271e-23	1624.5012664335666
cross-validation	1.8839335185148152e-14	91.34923571728213
simple-good-turing	4.030715115056194e-14	81.94394010897462

6 问题 3：整个测试集的困惑度

三种方法计算测试集的困惑度如11所示

6.1 结果分析

simple-good-turing 方法是最合理的，所以它产生的语言模型计算整个测试集的困惑度是最小的，其次是 cross-validation 方法，最差的是 add-one 方法。

表 9: " 相信" 位置上的预测概率最大的前 10 个词及其概率

词	laplace 概率	词	cross-validation 概率	词	simple-good-turing 概率
的	0.002963841138114997	的	0.10157624359206935	的	0.10706462727869298
不	0.0010451439802826568	不	0.03543352004379921	不	0.03690331852051876
在	0.0008891523414344991	在	0.03188389734133283	在	0.03120077155938853
想	0.0007487598664711572	想	0.026023549062332375	想	0.026069304578559525
是	0.000623966555392631	就	0.020737991247364255	是	0.02150909874535487
就	0.000623966555392631	是	0.01999395546162325	就	0.02150909874535487
，	0.0005615698998533679	，	0.017758561288546287	，	0.01922960878163332
和	0.0005615698998533679	和	0.017460698617354612	和	0.01922960878163332
也	0.0005303715720837364	也	0.016994234169977068	也	0.018090079922926834
要	0.0004991732443141048	要	0.01596836176415148	要	0.016950731037202276

表 10: " 相信" 的概率和位置

方法	概率	位置
laplace's law	0.0003119832776963155	18
cross-validation	0.009180154059163862	18
simple-good-turing	0.010121423202061827	18

表 11: 测试集的困惑度

方法	困惑度
laplace's law	7579.25835841848
cross-validation	937.1256035175825
simple-good-turing	912.2587057971734