通过本次试验完成了adding-one smoothing以及good-turing smoothing的unigram和bigram的模型计算句子概率，并完成了句子Perplexity的计算。

通过学习可知，Perplexity的计算公式为：

$$PP(S)=P(w_1w_2\ldots w_N)^{-\frac{1}{N}}$$

Perplexity就是在某种编码方式（语言模型）下评估测试集的平均编码长度，越小则代表模型散度越小，越贴近真是分布。易得句子S的出现的频度越高则PP(S)就会越低，通过实验得到结果如下(**表中数值为Perplexity均值**)：

	valid	**test**
Lap_Unigram	3789.42	3313.80
Lap_Bingram	1015270.78	960060.43
GT_Unigram	2198.79	2068.23
GT_Bingram	65808.52	64069.98

对于adding-one smoothing来说，由于train语料库中未出现的N-gram的数量太多，平滑后，所有未出现的N-gram占据了整个概率分布的很大比例，同理good-turing smoothing也存在这种问题，因为对原先概率为0的情况作了一刀切的处理，因此相较于Backoff取折扣因子递归寻找(n - 1)-gram，这两种的平滑效果要逊色很多。

假设语料中出现了r次的词有Nr个，相较于Laplace smoothing粗暴地对未出现的N-gram计1处理，Good turing smoothing考虑到r较小的时候，极大似然估计可能不准确，因此对所有r作了一个discount处理。显然后者性能要优于前者。分析数据，Unigram模式下，GT法比Lap法平滑效率高55%，而在Bingram模式下，GT法的与Lap法的效率天差地别，将近16倍。不难预测在Trigram或者Ngram的模式下GT法远优于Lap法。

本次实验结果基本符合预期，对Ngram模型评估语句有了进一步了解，遗憾由于时间限制，held-out和back-off未能完成。